



ARIB STD-B24
Version 5.0-E1

ENGLISH TRANSLATION

Data Coding and Transmission Specifications for Digital Broadcasting

ARIB STANDARD

ARIB STD-B24 Version 5.0

VOLUME 2 (1/2)

Established on	June	20, 2000	Version 1.0
Revised on	June	28, 2002	Version 3.2
Revised on	February	5, 2004	Version 4.0
Revised on	May	29, 2006	Version 5.0

Association of Radio Industries and Businesses (ARIB)

General Notes to the English translation of ARIB Standards and Technical Reports

1. The copyright of this document is ascribed to the Association of Radio Industries and Businesses (ARIB).
2. All rights reserved. No part of this document may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, without the prior written permission of ARIB.
3. The ARIB Standards and ARIB Technical Reports are usually written in Japanese and approved by the ARIB Standard Assembly. This document is a translation into English of the approved document for the purpose of convenience of users. If there are any discrepancies in the content, expressions, etc., between the Japanese original and this translated document, the Japanese original shall prevail.
4. The establishment, revision and abolishment of ARIB Standards and Technical Reports are approved at the ARIB Standard Assembly, which meets several times a year. Approved ARIB Standards and Technical Reports, in their original language, are made publicly available in hard copy, CDs or through web posting, generally in about one month after the date of approval. The original document of this translation may have been further revised and therefore users are encouraged to check the latest version at an appropriate page under the following URL:

<http://www.arib.or.jp/english/index.html>

Contents

Preface

Volume 1 Data Coding

Part 1 Reference Model for Data Broadcasting

Part 2 Monomedia Coding

Part 3 Coding of Caption and Superimpose

Volume 2 XML-based Multimedia Coding Scheme

Appendix 1 Operational Guidelines

Appendix 2 Operational Guidelines for Implementing Basic Services

Appendix 3 Operational Guidelines for Implementing Extended Services
for Fixed Receiving System

Appendix 4 Operational Guidelines for Implementing Extended Services
for Portable Receiving System

Appendix 5 Operational Guidelines for Implementing Extended Services
for Mobile Receiving System

Volume 3 Data Transmission Specification

Preface

ARIB (Association of Radio Industries and Businesses) establishes the "ARIB Standards" for the basic technical conditions of standard specifications related to variety of radio communication equipments, broadcasting transmission equipments, and its reception equipments using radio wave with the participation of radio communication equipment manufacturers, broadcasting equipment manufacturers, electric communication companies, service providers and other users.

"ARIB Standards" are nongovernmental standards established by combining governmental technical standards established for the purpose of effective use of frequency and to avoid interference of other users, and nongovernmental optional standards established for convenience for radio communication equipment manufacturers, broadcasting equipment manufacturers, electric communication companies, service providers and users, in order to secure appropriate quality and compatibility of radio communication equipment and broadcast equipment, etc.

This standard is established for "Data Coding and Transmission Specification for Digital Broadcasting" by the approval of the standardization committee, participated by radio communication equipment manufacturers, broadcast equipment manufacturers, electric communication companies, service providers and users irrespectively, to secure impartiality and clearness.

For data broadcasting of digital broadcasting, it is directed by the Telecommunications Technology Council on July 21, 1999 that it is desired that the most desirable multimedia coding specification in Japan at this point should be based on an XML-based specification, which is superior in many points such as "function", "contents production environment", "compatibility with other media", "data processing at terminal side", "extension ability of coding method", and "future direction of engineering development", etc., and that the detailed specifications should be standardized by the nongovernmental standardization organization with flexibility.

This standard is established as nongovernmental standard of data broadcasting specification used in Japan based on this direction, and consists of three parts: mono-media coding, multimedia coding, and data transmission specification. Compatibility with multiplex data broadcasting specification, which is already used in Japan is considered for mono-media coding. Compatibility with network usage or data broadcasting method in Europe and America is considered for multimedia coding and the coding scheme is based on XML coding specified in W3C specification adding necessary specifications for broadcasting. Each coding scheme in this standard is applied to whole broadcasting media generally and the conditions proper to broadcasting media derived from transmission methods and service requirements should be specified as operational restrictions.

Though this standard is mainly applied to BS digital broadcasting as the first step, the specification should be completed adding necessary specifications for other broadcasting media, considering trends of international standardization and new technological trends which cannot be assumed yet.

We hope that this standard will be put to practical use actively by radio communication equipment manufacturers, broadcast equipment manufacturers, electric communication companies, service providers, users, and so on.

Notice:

This standard does not describe industrial proprietary rights mandatory to this standard. However, the right proprietor of the industrial proprietary rights has expressed that "Industrial proprietary rights related to this standard, listed in the annexed table below, are possessed by the applicator shown in the list. However, execution of the right listed in the annexed table below is permitted indiscriminately, without exclusion, under appropriate condition, to the user of this standard. In the case when the user of this standard possesses the mandatory industrial proprietary rights for all or part of the contents specified in this standard, and when he asserts his rights, it is not applied."

Annexed table

Patent applicant	Name of invention	Patent number	Remarks
Matsushita Electric Industrial Co., Ltd.	情報処理装置	特開平 04-205415号	Japan
	データサーバ装置及び端末装置	特開平 06-139173号	Japan
	放送を用いて対話性を実現する送信装置、受信装置、受信方法、その受信プログラムを記録した媒体、通信システム	特開平 10-070712号	Japan
	データ入出力端末装置	特開平 10-074134号	Japan
	情報処理装置	特開平 10-083270号	Japan
	データの提示を制御するデータ提示制御装置、データの提示を～情報を送信するデータ送信装置及びデータ～データ提示制御情報編集装置	特開平 10-164530号	Japan
	デジタル放送システム、デジタル放送装置及びデジタル放送における受信装置	特開平 10-304325号	Japan
	デジタル放送装置、受信装置、デジタル放送システム、受信装置に適用するプログラム記録媒体	特開平 10-313449号	Japan
	番組編集装置および番組受信装置	特願平 10-020585号	Japan
	放送局システム及び受信機	特願平 10-195093号	Japan
	デジタル放送のための記録再生装置および方法	特願平 11-367308号	Japan
	データ送受信システムおよびその方法	特願平 11-103619号	Japan
	デジタルデータ送受信システムおよびその方法	特願平 11-124986号	Japan
	Submitted comprehensive confirmation of patents applied to the revised parts of ARIB STD-B24 Ver3.8 *5		
TOSHIBA CORPORATION	多重放送システムとこのシステムで使用される放送送信装置および放送受信装置	特開平 09-162821号	Japan

Patent applicant	Name of invention	Patent number	Remarks
	デジタル放送装置及びデジタル放送方法、デジタル放送受信装置及びデジタル放送受信方法、デジタル放送受信システム*16	特許第3621682号	Japan
NHK (Japan Broadcasting Corporation)	文書情報出力装置および方法	特開平 9-244617号	Japan
	入力データの自動選択処理装置	特開平 11-328189号	Japan
	マルチメディア型情報サービス方式およびその方式の実施に使用する装置	特開平 11-331104号	Japan
Sony Corporation *1	音声信号圧縮方法及びメモリ書き込み方法	特許第 1952835号	Japan
	オーディオ信号処理方法	特許第 3200886号	Japan
	オーディオ信号処理方法	特許第 3141853号	Japan
	信号符号化又は複合化装置、及び信号符号化又は複合化方法、並びに記録媒体	WO94/28633	Japan
	信号符号化方法及び装置、信号複合化方法及び装置、並びに記録媒体	特開平 7-168593	Japan
	符号化音声信号の複合化方法	特開平 8-63197	Japan
	音声信号の再生方法、再生装置及び伝送方法	特開平 9-6397	Japan
	音声信号の再生方法及び装置、並びに音声複合化方法及び装置、並びに音声合成方法及び装置、並びに携帯無線端末装置	特開平 9-190196	Japan
	音声符号化方法、音声複合化方法及び音声符号化複合化方法	特開平 8-69299	Japan
	音声符号化方法及び装置、音声複合化方法及び装置	特開平 9-127991	Japan
	符号化データ複合化方法及び符号化データ複合化装置	特許 2874745号	Japan
	映像信号符号化方法	特許 2877225号	Japan
	符号化データ編集方法及び符号化データ編集装置	特許 2969782号	Japan
	動画像データエンコード方法及び装置、並びに動画像データデコード方法および装置	特許 2977104号	Japan
	動きベクトル伝送方法及びその装置並びに動きベクトル複合化方法及びその装置	特許 2712645号	Japan
Mitsubishi Electric Corporation	Submitted comprehensive confirmation of patents applied to the revised parts of ARIB STD-B24 Ver3.1 *2		
	マルチメディア多重方式*3	特許第 3027815号	Japan

Patent applicant	Name of invention	Patent number	Remarks
	マルチメディア多重方式*3	特許第 3027816号	Japan
	Submitted comprehensive confirmation of patents applied to the revised parts of ARIB STD-B24 Ver4.4 *15		
Motorola Japan Ltd.	Submitted comprehensive confirmation of patents applied to the revised parts of ARIB STD-B24 Ver3.6 *4		
	Submitted comprehensive confirmation of patents applied to the revised parts of ARIB STD-B24 Ver3.8 *5		
	Submitted comprehensive confirmation of patents applied to the revised parts of ARIB STD-B24 Ver3.9 *6		
	Submitted comprehensive confirmation of patents applied to the revised parts of ARIB STD-B24 Ver4.0 *7		
	Submitted comprehensive confirmation of patents applied to the revised parts of ARIB STD-B24 Ver4.1 *9		
NTT DoCoMo, Inc. *11	動画像符号化方法、動画像複合方法、動画像符号化装置、及び動画像複合装置*11	特許第 3504256号	Japan, EPC, USA, Korea, China, Taiwan
	動画像符号化方法、動画像複合方法、動画像符号化装置、動画像複合装置、動画像符号化プログラム、及び動画像複合プログラム*11	特許第 3513148号	Japan, EPC, USA, Korea, China, Taiwan
	動画像複合方法、動画像複合装置、及び動画像複合プログラム*11	特許第 3534742号	Japan, EPC, USA, Korea, China, Taiwan
	信号符号化方法、信号複合方法、信号符号化装置、信号複合装置、信号符号化プログラム、及び、信号複合プログラム*11	特許第 3491001号	Japan, EPC, USA, Korea, China, Taiwan
	インターリーブを行うための方法および装置並びにデ・インターリーブを行うための方法および装置*13	特許第 3362051号	Japan, USA, Korea, Singapore, Australia, China

Patent applicant	Name of invention	Patent number	Remarks
	誤り保護方法および誤り保護装置*13	特許第 3457335号	Japan, USA, UK Korea, Germany, France Italy, Singapore, Australia, China
	Submitted comprehensive confirmation of patents applied to the revised parts of ARIB STD-B24 Ver4.4 *15		
Sharp Corporation *5	画像符号化装置および画像復号装置	特許第 2951861号	Japan
NEC Corporation *5	画像信号の動き補償フレーム間予測符号化・複合化方法とその装置	特許第 1890887号	Japan
	圧縮記録画像の再生方式	特許第 2119938号	Japan
	圧縮記録画像の対話型再生方式	特許第 2134585号	Japan
	適応変換符号化の方法及び装置	特許第 2778128号	Japan
	符号化方式および復号方式	特許第 2820096号	Japan
	変換符号化複合化方法及び装置	特許第 3070057号	Japan
	改良DCTの順変換計算装置および逆変換計算装置	特許第 3185214号	Japan
	適応変換符号化方式および適応変換複合方式	特許第 3255022号	Japan
Philips Japan, Ltd	Submitted comprehensive confirmation of patents applied to the revised parts of ARIB STD-B24 Ver4.0 *8		
	Submitted comprehensive confirmation of patents applied to the revised parts of ARIB STD-B24 Ver4.1 *10		
	Submitted comprehensive confirmation of patents applied to the revised parts of ARIB STD-B24 Ver4.2 *12		
Philips Electronics Japan, Ltd.	Submitted comprehensive confirmation of patents applied to the revised parts of ARIB STD-B24 Ver4.3 *14		

- Note) *1: valid for the revised parts of ARIB STD-B24 Ver3.0
 *2: valid for the revised parts of ARIB STD-B24 Ver3.1
 *3: valid for the revised parts of ARIB STD-B24 Ver3.3
 *4: valid for the revised parts of ARIB STD-B24 Ver3.6
 *5: valid for the revised parts of ARIB STD-B24 Ver3.8
 *6: valid for the revised parts of ARIB STD-B24 Ver3.9 (accepted on October 9,2003)
 *7: valid for the revised parts of ARIB STD-B24 Ver4.0 (accepted on January 8,2004)
 *8: valid for the revised parts of ARIB STD-B24 Ver4.0 (accepted on January 29,2004)
 *9: valid for the revised parts of ARIB STD-B24 Ver4.1 (accepted on November 17,2004)
 *10: valid for the revised parts of ARIB STD-B24 Ver4.1 (accepted on December 7,2004)

- *11: valid for the revised parts of ARIB STD-B24 Ver3.8 (accepted on January 7,2005)
- *12: valid for the revised parts of ARIB STD-B24 Ver4.2 (accepted on March 14,2005)
- *13: valid for the ARIB STD-B24 Ver1.0 (accepted on September 26,2005)
- *14: valid for the revised parts of ARIB STD-B24 Ver4.3 (accepted on September 27,2005)
- *15: valid for the revised parts of ARIB STD-B24 Ver4.4 (accepted on March 6,2006)
- *16: valid for the revised parts of ARIB STD-B24 Ver3.6 (accepted on March 14,2006)

VOLUME 2

XML-based Multimedia Coding Scheme

[BLANK]

Contents

Chapter 1 Purpose	1
Chapter 2 Scope	2
Chapter 3 Definitions and Terminology	3
3.1 Definitions	3
3.2 Terminology	3
Chapter 4 Coding of B-XML Documents	7
4.1 Character Code Sets	7
4.1.1 EUC-JP	7
4.1.2 JIS X 0221 (UCS)	8
4.1.3 Shift-JIS	8
4.2 XML Declaration	8
4.3 Document Type Declaration	9
4.4 System Identifier	9
4.5 Designation of a Style Sheet	9
4.5.1 Style sheet processing instructions	9
4.6 B-XML Version Information	10
4.7 Other XML Specifications	10
4.7.1 XML namespace	10
4.7.2 XML link language	10
4.7.3 XML pointer language	10
Chapter 5 BML: Application Language for Multimedia Presentation	11
5.1 Character Coding Schemes	11
5.1.1 Character coding schemes used for BML documents	11
5.1.2 Text modifier	11
5.1.3 External characters	11
5.2 Declarations in a BML Document	12
5.2.1 XML declaration	12
5.2.2 DTD file name	12
5.2.3 Version information of coding scheme	12
5.3 BML Elements	13
5.3.1 Core modules	13
5.3.2 Text Extension modules	13
5.3.3 Basic Forms module and Forms module	14
5.3.4 Basic Table module and Tables module	14
5.3.5 Image module	14
5.3.6 Client-side Map module	14
5.3.7 Server-side Map module	15
5.3.8 Object module	15

5.3.9	Frames module	15
5.3.10	Target module	15
5.3.11	Iframe module	15
5.3.12	Intrinsic Events module	15
5.3.13	Metainformation module.....	15
5.3.14	Scripting module	16
5.3.15	Style Sheet module.....	16
5.3.16	Style Attribute module	16
5.3.17	Link module	16
5.3.18	Base module.....	16
5.3.19	Name Identification module.....	16
5.3.20	Extension modules(BML/Basic BML modules).....	16
5.4	CSS-based Style Sheet	29
5.4.1	Media type.....	29
5.4.2	Box model.....	30
5.4.3	Visual formatting model	32
5.4.4	Other visual effects	32
5.4.5	Paged media	33
5.4.6	Colors and backgrounds.....	33
5.4.7	Fonts.....	35
5.4.8	Text	35
5.4.9	Pseudo classes and pseudo elements.....	35
5.4.10	Table related properties.....	35
5.4.11	User interface	35
5.4.12	Aural style sheet.....	36
5.4.13	Extended properties.....	36
Chapter 6	Converting XML Document into BML Using XSL	40
6.1	Structure of XSL Documents	40
6.2	XSLT Specifications	40
6.2.1	XSLT.....	40
6.2.2	Character encoding	40
6.2.3	Numbers	40
6.2.4	XSLT style sheet elements.....	41
Chapter 7	Procedural Description Language.....	44
7.1	DOM API.....	44
7.1.1	Core DOM Fundamental Interfaces	44
7.1.2	Core DOM Extended interfaces	44
7.1.3	HTML DOM interfaces	44
7.1.4	CSS DOM interface	45
7.1.5	Event DOM interface	50
7.1.6	BML extended DOM interface	53
7.2	Scripting Language	66
7.2.1	Base conventions.....	66

7.2.2 Additional conventions	67
7.2.3 Language binding.....	67
7.3 Security for Content	67
7.4 Native Objects.....	68
7.5 Extended Object for Broadcasting	68
7.5.1 CSVTable object.....	68
7.5.2 BinaryTable object.....	73
7.5.3 XML document Object	79
7.6 Extended Functions for Broadcasting (Browser Pseudo Object).....	82
7.6.1 EPG functions	82
7.6.2 Event group index functions	87
7.6.3 Series reservation functions	90
7.6.4 Subtitle presentation control functions.....	92
7.6.5 Non-volatile memory functions	95
7.6.6 Extended APIs for Storing	102
7.6.7 Interaction Channel functions	118
7.6.8 Operational control functions.....	148
7.6.9 Receiver sound control.....	165
7.6.10 Timer functions	165
7.6.11 External character functions.....	168
7.6.12 Functions for controlling external devices.....	169
7.6.13 Functions for controlling bookmark areas	171
7.6.14 Other functions.....	175
7.6.15 Ureg pseudo object properties.....	177
7.6.16 Greg pseudo object properties.....	177
7.6.17 Functions for Printing	177
7.7 Navigator pseudo object properties.....	206
7.8 Functions for interoperability with JavaScript.....	207
7.9 Security Class for content and Extended Functions for Broadcasting	207
Chapter 8 Monomedia Coding Schemes and Transmission Used in BML/B-XML Documents.....	214
8.1 Video Coding Scheme and Transmission	214
8.1.1 Transmission of MPEG-1 video.....	214
8.1.2 Transmission of MPEG-2 video.....	214
8.1.3 Transmission of MPEG-4 video and H.264 MPEG-4 AVC	214
8.1.4 Transmission of MPEG video/audio in a time-stamped TS format	215
8.2 Coding Scheme and Transmission of Still Pictures and Bitmap Graphics	218
8.2.1 Transmission of MPEG-2 I-frame, MPEG-4 I-VOP, and H.264 MPEG-4 AVC I- picture	218
8.2.2 Transmission of JPEG still picture.....	220
8.2.3 Coding scheme and transmission of PNG bitmap.....	220
8.2.4 Coding scheme and transmission of MNG animation	221
8.3 Audio Coding Scheme and Transmission	221
8.3.1 Transmission of MPEG-2 audio.....	221

8.3.2	Transmission of MPEG-4 audio.....	221
8.3.3	Transmission of PCM (AIFF-C) audio	222
8.3.4	Transmission of Additional Sound.....	222
8.4	Character Coding and Transmission	222
8.4.1	Transmission of EUC-JP text.....	222
8.4.2	Transmission of UCS/UTF-16 text	222
8.4.3	Transmission of Shift-JIS text.....	222
8.4.4	Transmission of 8-bit character code text	222
8.5	Graphic Description Command Coding Scheme and Transmission	222
8.5.1	Transmission of Geometric graphic data	222
8.6	External Font Coding Scheme and Transmission	223
8.6.1	Transmission of DRCS	223
8.7	Transmission of Two-dimensional Table Data	223
8.7.1	Transmission of table data handled by a CSVTable object	223
8.7.2	Transmission of table data handled by a BinaryTable object	223
8.8	Transmission of External XML Document.....	223
8.9	Transmission of Proprietary Data	223
8.9.1	Transmission in ES with B-XML/BML Data Coding Identification.....	223
8.9.2	Transmission in Independent ES.....	223
Chapter 9	Content Transmission and Namespace.....	224
9.1	Transmission of Content	224
9.1.1	Transmission in data carousel	224
9.1.2	Resource-to-module mapping.....	224
9.1.3	Transmission of event messages	233
9.2	Namespace	234
9.2.1	Component structure and resource namespace in event.....	234
9.2.2	Startup BML/B-XML document.....	236
9.2.3	Reference of AV streams and subtitle component.....	236
9.2.4	Identification of MPEG-I frames transmitted through a still picture carousel.....	237
9.2.5	Service identification	237
9.2.6	Event identification.....	238
9.2.7	Identification of stored contents.....	238
9.2.8	Identification of ERT node of Local Event Indexes	238
9.2.9	Names of sound built in the receiver.....	238
9.2.10	Namespace of persistent memory	238
9.2.11	Identification of component ES transmitting event message	239
9.2.12	Identification of series.....	239
9.2.13	Namespace used for obtaining resources through an IP packet transmission line	240
9.2.14	Data storage and Name descriptor values	240
9.2.15	Namespace convention for file in storage media	240
9.2.16	Namespace and reference convention for time-stamped TS format AV file and stored TS file	243
9.2.17	Namespace for external devices.....	244

9.3 Data Structure of PSI/SI Descriptor That Depends on Transmission of B-XML/BML	
Contents	245
9.3.1 Identification of data coding scheme	245
9.3.2 Information encoded in additional identification information area of data coding scheme descriptor	245
9.3.3 Information encoded in selector area of data contents descriptor	248
9.3.4 Information to be written in private area of DII	251
9.3.5 Descriptor in the module information area of DII	252
Chapter 10 XHTML-based BML Encoding using XML Namespace	254
10.1 XML Namespace	254
10.2 BML Encoding and XML Namespace	254
Annex A Coding Schemes of Color Map Data	256
Annex B Coding Schemes for Designation of Regions Using Zip Code	259
B.1 Overall Structure	259
B.2 Base Format	259
B.3 Examples	260
Annex C Media Type of B-XML/BML Documents and Monomedia Data	261
Annex D Document Type Definition of BML	264
D.1 BML Driver DTD	264
D.2 BML Extension Elements DTD	271
D.3 Basic BML Extension Elements Module DTD	274
D.4 BML Document Model Module	277
D.5 BML qname Module	282
Annex E Resource List for Content to Be Received in Real Time	288
Informative Explanation	294
1 Relationship between B-XML Architecture and Multimedia Description Language BML, and Guarantee of Future Evolution	294
2 Audio Playback Control	295
3 Multiplexing of Still Picture Carousels and Receiver Operation	296
4 Name sharability between real-time data services and stored data services	297
5 Sample of controlling external device by using External XML document	299
6 Overview of Bookmark	300
7 Access-controlled area and non-access-controlled area in non-volatile memory	302
References	305

[BLANK]

Chapter 1 Purpose

The standard described in Volume 2 provides an XML-based multimedia coding scheme specification for the data broadcasting scheme, part of the digital broadcasting scheme specified as the standard in Japan.

Chapter 2 Scope

The standard described in Volume 2 is applied to XML-based multimedia coding for data broadcasting carried out as part of digital broadcasting.

Chapter 3 Definitions and Terminology

3.1 Definitions

BML (Broadcast Markup Language): The XML application language specified in this standard to be solely responsible for tags and attributes for multimedia representation.

B-XML (Broadcast XML): The XML architecture in which an XML tag is defined by each application with a DTD specific to the application and converted into a BML tag by XSLT before to be presented on a terminal.

reserved: The term "reserved", when used in the clauses defining the coded bit stream, indicates that the value may be used in the future for ISO defined extensions. Unless otherwise specified within this standard, all reserved bits must be set to 1.

reserved_future_use: The term " reserved_future_use ", when used in the clauses defining the coded bit stream, indicates that the value may be used in extensions defined by ISO in the future. Unless otherwise specified within this standard, all reserved bits must be set to 1.

3.2 Terminology

attribute: A parameter to represent the character of a property.

automatic connection: A feature that establishes a connection upon a request in an IP packet using a receiver's ability, instead of an explicit function call used by a content to establish a connection. More specifically, a connection is established based on a configuration for ISP connection held by a receiver. Note that a receiver defaults to this automatic connection

BASIC mode data transmission control procedure: A communications protocol developed for basic data transmission between a host and a terminal. The protocol employs a method for minimizing transmission errors.

BML content: A set of information that consists of a BML document and a group of data including monomedia accompanying the BML document.

broadcaster-specific root certificate: A root certificate that is transmitted through a data carousel which is to be used temporarily by a receiver.

Browser pseudo object: The objects added to realize functions unique to broadcasting. Unlike a Native Object, they do not inherit properties.

built-in object: An object which is implemented in the ECMAScript execution system from the start of a script execution. There are nine types of objects (Array, Boolean, Date, Function, Global, Math, Number, Object, String).

call: To transmit a signal in order to establish a telecommunication channel from a device including a telephone.

CAS (Conditional Access System): Conditional access system.

CLUT (Color Look Up Table): A table used to convert an index color value to a physical value.

constructor: A function which generates and initializes an object.

CSS (Cascading Style Sheets): A standard describing style sheets for a HTML document.

data carousel: A method that sends out any set of data repeatedly so that the data can be downloaded via broadcasting in any timing as needed. This method is defined in ISO/IEC 13818-6.

DHCP: (Dynamic Host Configuration Protocol [RFC 2131]): A protocol used to automatically configure terminals on a TCP/IP network. For example, this protocol allows IP addresses to be assigned dynamically.

DNS (Domain Name System): A protocol used by the service that maps a host name on a network into its IP address.

DOM (Document Object Model): An API, also known as DOM-API, that defines the logical structure of an XML or HTML document and the way to access or manipulate an XML or HTML document. This API is an independent interface of platforms and languages.

DOM object: An object generated by a BML document.

DRCS (Dynamically Redefinable Character Sets): A scheme used to transmit a dynamically redefinable external character to receivers or other devices using the corresponding pattern.

DSM-CC (Digital Storage Media Command and Control): The Control method defined in ISO/IEC 13818-6, which provides access to a file or stream in digital interactive services.

DTD (Document Type Definition): A declaration that describes a document type used for XML.

ECMAScript: The programming language defined in the ECMA-262 standard.

EIT (Event Information Table): A table that contains data concerning an event, or program such as an event name, a start time, and a duration.

element: A document structuring unit delimited by tags. An element is delimited by a start-tag and an end-tag, except an empty element that is delimited by an empty-element tag.

entity: A piece of information which is transmitted as a result of a request or a response. It consists of an entity header field containing metadata and an entity body conveying content.

EPG (Electronic Program Guide): A program table that is presented electronically.

ES (Elementary Stream): A basic stream that contains video data, audio data, or private data. A single elementary stream is carried in a sequence of PES packets with one and only one stream_id.

Ethernet: ([IEEE 802]) A LAN standard that defines a bus-based network employing the CSMA/CD (Carrier Sense Multiple Access/Collision Detection) for access control.

EUC-JP (Extended Unix Code (-JP)): A Japanese character code used predominantly in a UNIX environment. It is encoded based on ISO-2022.

event: A collection of the broadcast data stream structuring units, elementary streams, that comprise a single service, representing a single program including a news program and a drama program. Each elementary stream is preconfigured with a start time, and an end time common to the service.

event handler: A user defined function which is triggered by a key input or an event invoked by a transmitted signal.

extended object for broadcasting: The additional ECMAScript objects which are specified in this standard to extend ECMAScript. The two types of objects, the CSVTable object and BinaryTable object are specified. Like a Native object, they inherit properties.

field: An element in a two-dimensional binary data table.

FTP (File Transfer Protocol [RFC959]): A protocol to share and transfer a file between two hosts via TCP/IP.

generic root certificate: A root certificate that is transmitted through a data carousel which is to be stored in a receiver. A receiver refers to a generic root certificate to establish an encrypted transmission.

generic root certificate storage area: An area that is in NVRAM or other types of memory in a receiver and is used to store generic root certificates.

HTTP (Hyper Text Transfer Protocol [RFC2068]): An application layer protocol used to transmit data over the World Wide Web.

IDL (Interface Definition Language): A language used to define the interface to access to and operate objects.

inheritance: The ability to create a new interface that has the methods and properties of its parent interface. instance: A substance of an object.

IP: (Internet protocol [RFC 791]) A network layer protocol that defines the addressing mechanism on the Internet to allow data to be transmitted.

language binding: A specification for binding a DOM API to a programming language. In BML, a DOM API is bound to ECMAScript. The term ‘property’ used in ECMAScript has the same meaning as the term ‘attribute’ used in IDL has.

local event: A subdivision of an event, which may be subdivided according to the temporal axis, by program component or based on others.

mail address: A representation of a location to which a mail is sent with SMTP. It has a style of “addressee@domain”. A domain corresponds to a domain name identified by a DNS.

method: A property of an object. More technically, a method is a function that is associated with an object and is allowed to manipulate the object's data.

MIME (Multipurpose Internet Mail Extensions): An application layer protocol that provides a content architecture that allows multimedia data such as non-US-ASCII format text files, audio files, and image files to be transmitted via Internet e-mail.

module: A unit of data transmitted with the data carousel scheme. A module consists of blocks (DownloadDataBlocks).

multipart format: An entity which has a single entity body consisting of two or more encapsulated entities.

name server: An administration name server based on DNS(Domain Name System). In general, a name server is a server machine which administrates correspondence between names of terminals and IP addresses.

NaN(Not-a-Number): A special value used in ECMAScript, which means that the concerned representation is not a number.

Native object: An object unique to ECMAScript. There are two types of native objects: built-in objects and objects generated while ECMAScript is executed.

node: A branch point of a tree consisting of generated DOM objects. A unique node which is not a child of any other node of the generated tree is called the root node. A node which is a parent of other node is called a parent node. Nodes which have the same parent are called sibling nodes. A node which is a child of other node is called a child node.

The word *node* used to describe event group index functions has another meaning. A node is a branch of a tree representing relationships among events, local events, and others, and encoded as extended information of SI.

NPT (Normal Play Time): An absolute temporal coordinate which represents the position in a stream at which an event is occurred.

object: A collection of named pieces of data. Each named piece of data is referred to as a property (See property).

PPP (Point to Point Protocol[RFC1661]): A protocol designed to transfer multiple protocols via a point-to-point linkage. PPP is used for dial up connections.

PPoE(PPP over Ethernet [RFC 2516]): A protocol that enables PPP frames to be transmitted over an Ethernet network.

property: A piece of information which is contained in a object. The properties of ECMAScript are the five types of primitive values (number, string, boolean, null and undefined), objects and methods.

prototype: An object used for sharing or inheriting a property.

PSI (Program Specific Information): A piece of information to control transmission, as defined in ISO/IEC13818-1. PSI consists of normative data which is necessary to demultiplex multiplexed Transport Streams and to regenerate programs as intended.

record: A row of a two-dimensional binary data table.

resource: A network data object or a service which is uniquely identified in a network.

scripting language: A language used to describe a program process which is built in BML documents.

SKC(shared key cryptosystem): A cryptosystem also known as a secret key cryptosystem or a symmetric key cryptosystem. In this system, a sender encodes a message using a secret key and the intended receiver decodes the sent message using the same key that is shared and kept private by the sender and receiver. A separate scheme that enables the sender and the receiver to share the secret key is required.

Shift-JIS: A Japanese character code used predominantly in a PC environment. This character encoding is defined in Appendix 1 of JIS X0208:1997

section: A section is a syntactic structure specified in ISO/IEC13818-1. It is used for mapping service information, as defined in ARIB STD-B10, into an ISO/IEC 13818-1 Transport Stream packet.

service: A sequence of programs that is organized by a broadcaster and can be broadcast as part of a schedule.

SI (Service Information): Digital data that describes an arrangement of programs, as specified in ARIB STD B-10, which has been developed based on the DVB-SI specification.

sibling nodes: Nodes which have the same parent.

SMTP (Simple Mail Transfer Protocol [RFC821]): A protocol used to relay and deliver e-mails.

SSL (Secure Socket Layer): A security protocol that works at a socket level. This layer exists between the TCP layer and the application layer to encrypt/decode data and authenticate concerned entities.

TCP(Transmission Control Protocol [RFC 793]): A transport layer protocol that provides highly reliable end-to-end, connection-oriented data delivery using an error detection and correction mechanism.

TLS(Transport Layer Security [RFC 2246]): A protocol used to send and receive encoded data over the Internet. This protocol supports a combination of various security technologies including PKC, SKC, digital certificates, hash functions to prevent eavesdropping, message forgery, and spoofing.

UCS (Universal (Coded) Character Set): An international character code set developed by ISO (ISO/IEC 10646).

URI (Uniform Resource Identifier): An addressing method to access to objects in the Internet.

UTF (UCS Transformation Format): A transformation method for UCS.

variable argument list: A group of arguments in the case where number of the arguments for a function is not constant.

X.28: An ITU-T Recommendation which defines the function to access from a public subscriber telephone network to a public data network through packet switching.

XHTML (eXtensible HTML): An extended version of HTML. In the XHTML specification, an HTML document is recognized as an XML application.

XSL (eXtensible Stylesheet Language): The style sheet recommendation for XML.

Chapter 4 Coding of B-XML Documents

This chapter defines the architecture of B-XML. The B-XML architecture uses a pair of tags whose semantics are specific to an application to describe data as intended. The B-XML architecture converts an XML document that conforms to XML1.0, a W3C Recommendation, into a BML document using XSLT for presentation. This chapter also provides additional definitions of XML1.0 that are required to implement the B-XML architecture.

4.1 Character Code Sets

A B-XML document uses one of the following character encoding schemes:

- EUC-JP
- JIS X 0221-1995 (UCS)
- Shift-JIS

Not more than one scheme must be used in any single document and any external data referenced by the document.

4.1.1 EUC-JP

EUC-JP is defined as a variation of the 8-bit character code set specified in ARIB STD-B24 Volume 1 Part2. EUC-JP works the same as the 8-bit character code that invokes G0 set to GL and G1 set to GR, except that EUC-JP does not use control codes assigned to C0 and C1 control code areas. Table 4-1 illustrates the EUC-JP code set. The OVERLINE (its character code is 0x7E) defined in JIS X 0201 is expressed with “~”, and mapped to TILDA in the unicode character set. The character shape implemented in a receiver may be a wavy line or a straight line.

Table 4-1 EUC-JP Code Set

Code Set	Character Set	Remarks
Code Set 0 Byte range: 21~7E	JIS X 0201-1976 (JIS Roman Characters)	Equivalent to ARIB-STD-B5 Alphanumeric Set
Code Set 1 First byte range: A1-FE Second byte range: A1-FE	JIS X 0208-1990 (Those of ARIB-STD-B5 Kanji Set is allocated to 90 to 94 Ku (Free Area).)	Equivalent to ARIB-STD-B5 Kanji Set
Code Set 2 First byte: 8E Second byte range: A1-DF	JIS X 0201-1997 (Halfwidth Katakana)	
Code Set 3 First byte: 8F Second byte range: A1-FE Third byte range: A1-FE	JIS X 0212-1990	
Control Codes	Space character (20)	Equivalent to ARIB-STD-B5 SP
	Delete character (7F)	
	New Line (0D0A)	
	Tab (09)	
	SS2 (8E)	
	SS3 (8F)	

4.1.2 JIS X 0221 (UCS)

The character set and the encoding scheme for UCS comply with the definitions in ARIB STD-B24 Volume 1 Part 2.

However, when the JIS X 0221 character set is used in a B-XML document, the following encoding scheme must be used.

- UTF-16 is used as a character encoding scheme.
- UTF-16 data is transmitted in the big endian byte order, i.e. the most significant byte is transmitted first.

4.1.3 Shift-JIS

The character set of Shift JIS is shown in Table 4-2, as specified in Appendix 1 of JIS X0208:1997. Note that the Kanji characters in the range from 90 Ku (block) to 94 Ku (block) specified in Volume 1 are used. The OVERLINE (its character code is 0x7E) defined in JIS X 0201 is expressed with “~”, and mapped to TILDA in the unicode character set. The character shape implemented in a receiver may be a wavy line or a straight line.

Table 4-2 Shift JIS Code Set

Code Set	Character Set	Remarks
Single-byte (Halfwidth) Characters Byte range: 21~7F, A1~DF	JIS X 0201-1997 (JIS Roman Characters) JIS X 0201-19767 (Halfwidth Katakana)	
Double-byte Characters First byte range: 81~9F, E0~EF Second byte range: 40~7E, 80~FC	JIS X 0208-1997 (Those of ARIB-STD-B5 Kanji Set is allocated to 90 to 94 Ku (Free Area).)	
Control Codes	Space character (20)	
	Delete character (7F)	
	New Line (0D0A)	
	Tab (09)	

Shift-JIS is converted into EUC-JP as the following:

The characters 0x00-0x7F in Shift-JIS is converted into the characters 0x00-0x7F in the Code Set 0 in EUC-JP, respectively.

The characters 0xA1-0xDF in Shift-JIS is converted into the characters 0x8EA1-0x8EDF in the Code Set 2 in EUC-JP, respectively.

The characters 0x8140-0xEFFC in Shift-JIS is converted into the characters 0xA1A1-0xFEFE in the Code Set 1 in EUC-JP, respectively.

4.2 XML Declaration

In the case of using the JIS X 0221 character encoding scheme, it must be described as UTF-16 in the XML declaration, as follows:

```
<?xml version="1.0" encoding="UTF-16"?>
```

4.3 Document Type Declaration

When a B-XML document is not a valid document, the B-XML document must be a well-formed document that may have no DTD. A valid B-XML document must contain a DTD, as specified in the XML specifications.

Example: A typical document type declaration is shown in the following. (The root element type is indicated as xxx and URI as system identifier ###.)

```
<!DOCTYPE xxx SYSTEM "###" >
```

4.4 System Identifier

In the W3C XML 1.0 specification, system identifiers are used in the Document Type, Entity and Notation declarations. URIs that are used as system identifiers conform to the namespace of resources defined in BML.

Example: Document Type, Entity and Notation declarations using system identifiers are shown in the following.

```
<!DOCTYPE yyyy SYSTEM "arib-dc://.../hello.dtd">  
<!ENTITY zzzz SYSTEM "arib-dc://.../hello.xml">  
<!NOTATION xxxx SYSTEM "arib-dc://.../hello.exe" >
```

4.5 Designation of a Style Sheet

The method in which an XML document refers to an XSL document conforms to the W3C Recommendation, Associating Style Sheets with XML documents.

4.5.1 Style sheet processing instructions

XSL documents can be referenced to by writing style sheet processing instructions in an XML document.

Style sheet processing instructions define the following pseudo attributes.

- (1) href
Conforms to the BML namespace definitions. Only XSL style sheet can be specified. Direct reference to CSS is not supported.
- (2) type
The available MIME type is limited to text/xsl.
- (3) title
Conforms to HTML4.0 definitions.
- (4) media
Conforms to HTML4.0 definitions.
- (5) charset
Charset attribute is the same as those in the referencing XML document.
- (6) alternate
Conforms to the HTML4.0 definitions.

Example: A style sheet processing instruction is shown in the following.

```
<?xml-stylesheet href="mystyle.xsl" type="text/xsl" media="tv"?>
```

4.6 B-XML Version Information

The following processing instructions must be written in a B-XML document. They identify BML documents that contain only the elements defined in Chapter 5 of this document, and the B-XML version to which the document conforms.

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

The version number consists of a major number and a minor number. The range of major number is 1 to 255 and the range of minor number is 0 to 255. The numbers are represented as a decimal number character string with leading zeros suppressed. The initial standard version number is 1.0.

If receivers that conform to the older versions can still receive a document based on the revised specifications in relation to error corrections or operational reasons, minor number must be updated. If receivers that conform to the older versions cannot receive a document based on the revised specifications, major number must be updated.

4.7 Other XML Specifications

4.7.1 XML namespace

The specifications for the B-XML namespace conform to the W3C Recommendation, "Namespaces in XML." However, the following restrictions apply to the leading character of XML namespace prefixes.

XML namespace prefixes starting with a character string "bxml" (case-insensitive) must be reserved for this specification and for its future extensions.

If the XML name spaces are used in the BML defined in the chapter 5, those are defined in the chapter 10.

4.7.2 XML link language

B-XML documents that conform to this Standard must not use XLink defined in the W3C Recommendation, "XML Linking Language (XLink)." The linking specifications in BML must be used for linking.

4.7.3 XML pointer language

B-XML documents that conform to this Standard must not use XPointer defined in the W3C Recommendation, "XML Pointer Language (XPointer)."

Chapter 5 BML: Application Language for Multimedia Presentation

This chapter specifies an XML application language called “BML”, which is used for multimedia presentation. This language is based on XHTML1.0, CSS1, and a part of CSS2 that are defined by W3C. BML employs ECMAScript as a script description language and also has functional extensions that are required for broadcasting services.

5.1 Character Coding Schemes

5.1.1 Character coding schemes used for BML documents

A BML document uses one of the following character encoding schemes:

- EUC-JP
- JIS X 0221 (UCS)
- Shift-JIS

Note: Only one scheme must be used in any single BML document and any external data including ECMAScript files and CSS files referenced by the document.

5.1.1.1 EUC-JP

The conventions in Section 4.1.1 is applied to any BML document described with EUC-JP.

When a BML document is written in EUC-JP, any external file that is referenced with an object element may contain 8-bit character codes including control codes and external characters.

5.1.1.2 JIS X 0221 (UCS)

BML documents written in the UCS character set must conform to the conventions in Section 4.1.2.

Note that any external file that is referenced with an object element by a BML document written in the UCS character set may contain 8-bit character codes including control codes and external characters.

5.1.1.3 Shift-JIS

BML documents written in the Shift-JIS character set must conform to the conventions in Section 4.1.3.

Note that any external file that is referenced with an object element by a BML document written in the Shift-JIS character set may contain 8-bit character codes including control codes and external characters.

5.1.2 Text modifier

Text modifiers in a BML document is implemented using inline elements defined in Section 5.3.2.1 and CSS defined in Section 5.4. For details of the text modifier functionality, see Section 5.4.

5.1.3 External characters

The character coding scheme of external characters conforms to DRCS that is defined in ARIB STD-B24 Volume 1 Part 2.

External characters in a BML document become effective only after external DRCS data is explicitly loaded with the loadDRCS() function in ECMAScript. The external characters loaded with the loadDRCS() function are effective only in that BML document.

For an external 8-bit code text file referenced from a BML document, the DRCS data may be applied according to the definition of the 8-bit encoding scheme. It must be defined in an operational standard regulation whether or not an external 8-bit code text file can use external characters loaded with the loadDRCS() function.

5.2 Declarations in a BML Document

5.2.1 XML declaration

The XML version in an XML declaration must be 1.0. For BML documents written in EUC-JP, the character encoding identifier must be set to EUC-JP.

Example:

```
<?xml version="1.0" encoding="EUC-JP" ?>
```

For BML documents written in JIS X 0221, the character encoding identifier shall be set to UTF-6.

Example:

```
<?xml version="1.0" encoding="UTF-16" ?>
```

For BML documents written in Shift-JIS, the character encoding identifier must be set to Shift_JIS.

Example:

```
<?xml version="1.0" encoding="Shift_JIS" ?>
```

5.2.2 DTD file name

The name of a DTD file conforms to the following convention that uses major number and minor number in the version information defined in the BML specifications.

```
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 number are not part of the coding scheme version described in the next section.

5.2.3 Version information of coding scheme

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

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 BML document in an extended coding scheme can be successfully viewed with older BML browsers, the minor version number must be updated and the major version number must not be updated. When a BML document in an extended coding scheme cannot be successfully viewed without a newer BML 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]" ?>

5.3 BML Elements

This section defines the elements that can be used in a BML document.

The elements and attributes conform to the Strict document type definition that is defined in XHTML1.0. Any modularization conforms to the “Modularization of XHTML” W3C Recommendation. The following description defines the elements and attributes that are used in BML for each XHTML module. The Applet module, the Name Identification module, and the Legacy module must not be applied to any BML document.

The DTD of a BML document is defined in Annex C.

5.3.1 Core modules

The modules that define elements are described in Sections 5.3.1.1 - 5.3.1.4.

5.3.1.1 Structure module

This module defines elements for indicating the major structure.

The elements include the body element and the html element.

This module complies with Section 5.2.1 in “Modularization of XHTML”.

5.3.1.2 Text module

This module defines elements for presenting text of a document.

The elements include the br element, the h1-h6 elements, the pre element, and the span element.

This module complies with Section 5.2.2 in “Modularization of XHTML”.

5.3.1.3 Hypertext module

This module defines an element for specifying hypertext links to other BML documents. The module consist of the a element.

This module complies with Section 5.2.3 in “Modularization of XHTML”.

5.3.1.4 List module

This module defines elements for providing list-style presentations.

The elements include the dl element and the ol element.

This module complies with Section 5.2.4 in “Modularization of XHTML”.

5.3.2 Text Extension modules

The modules used to add textual presentations are described in Sections 5.3.2.1 - 5.3.2.3.

5.3.2.1 Presentation module

This module defines elements for using text modifiers and character styles including a bold type.

The elements include the **b** element and the **hr** element.

This module complies with Section 5.4.1 in “Modularization of XHTML”.

5.3.2.2 Edit module

This module defines elements for editing a BML document.

The module consists of the **del** element and the **ins** element.

This module complies with Section 5.4.2 in “Modularization of XHTML”.

5.3.2.3 Bi-directional Text module

This module defines elements for controlling the direction of textual presentations in a BML document.

The module consists of the **bdo** element.

This module complies with Section 5.4.3 in “Modularization of XHTML”.

5.3.3 Basic Forms module and Forms module

These modules define elements for controlling interactive data input operations.

The elements include the **input** element and the **textarea** element.

This module complies with Sections 5.5.1 and 5.5.2 in “Modularization of XHTML”. Implementing this module involves ensuring security necessary to send, receive, or use information inputted with this module, including passwords or other private information.

5.3.4 Basic Table module and Tables module

These modules define elements for providing table-style presentations.

The elements include the **table** element.

This module complies with Sections 5.6.1 and 5.6.2 in “Modularization of XHTML”.

5.3.5 Image module

This module defines an element for embedding images in a BML document.

The module consists of the **img** element.

This module complies with Section 5.7 in “Modularization of XHTML”.

5.3.6 Client-side Map module

This module defines elements for ensuring image mapping that is responsible for a terminal, or client.

The elements include the **area** element and the **map** element.

This module complies with Section 5.8 in “Modularization of XHTML”.

5.3.7 Server-side Map module

This module defines elements for ensuring image mapping that is responsible for a server.

The module consists of the `img&` element and the `input&` element.

This module complies with Section 5.9 in “Modularization of XHTML”.

5.3.8 Object module

This module defines elements for generic objects that represent images, video, and audio.

The module consists of the `object` element and the `param` element.

This module complies with Section 5.10 in “Modularization of XHTML”.

5.3.9 Frames module

This module defines elements for frame-style presentations.

Although the Frames module is not part of the Strict document type, this module is included in the BML definitions.

The elements include the `frame` element and the `frameset` element.

This module complies with Section 5.11 in “Modularization of XHTML”.

5.3.10 Target module

This module defines attributes for describing target-related information.

The elements include the `target` element.

This module complies with Section 5.12 in “Modularization of XHTML”.

5.3.11 Iframe module

This module defines elements for inserting frames into text.

The module consists of the `iframe` element.

This module complies with Section 5.13 in “Modularization of XHTML”.

5.3.12 Intrinsic Events module

This module defines attributes that correspond to events generated by user operation.

The attributes include the `onclick` attribute.

This module complies with Section 5.14 in “Modularization of XHTML”.

5.3.13 Metainformation module

This module defines elements for presenting meta information of a document.

The module consists of the `meta` element.

When the meta element is used to control behaviours of a BML browser, a string that begins with “ARIB” must be used as a value for the name attribute of the meta element.

This module complies with Section 5.15 in “Modularization of XHTML”.

5.3.14 Scripting module

This module defines elements for scripts that describe behaviours and elements for controlling scripts.

The module consists of the script element and the noscript element.

This module complies with Section 5.16 in “Modularization of XHTML”.

5.3.15 Style Sheet module

This module defines elements for describing style sheets.

The module consists of the style element.

This module complies with Section 5.17 in “Modularization of XHTML”.

5.3.16 Style Attribute module

This module defines the style attribute.

This module complies with Section 5.18 in “Modularization of XHTML”.

5.3.17 Link module

This module defines an element for providing document-related information for a browser.

The module consists of the link element.

This module complies with Section 5.19 in “Modularization of XHTML”.

5.3.18 Base module

This module defines an element for defining a base URI.

The module contains the base element.

This module complies with Section 5.21 in “Modularization of XHTML”.

5.3.19 Name Identification module

The module defines the name attribute.

This module complies with Section 5.20 in “Modularization of XHTML”.

5.3.20 Extension modules(BML/Basic BML modules/Basic Mobile BML/Server-based BML module)

BML has the following extension modules to define the following elements and attributes.

The four modules are designed to define additional features specific to BML. The Basic BML module is limited to the basic features. The BML module supports the necessary features. The Basic Mobile

BML module has features for mobile receivers. The server-based BML module is designed to cover server-based broadcasting.

The Basic BML module, the Basic Mobile BML module, and the server-based BML module are subsets of the BML module. Not more than one of the four modules may be applied to a single document type. The XML namespace must be used to differentiate the four extension modules from other XHTML modules.

Table 5-1 Basic BML module

Element	Attribute	Minimum content model
bevent	id (ID)	(beitem)+
beitem	id (ID), type ("EventMessageFired" "EventFinished" "EventEndNotice" "Abort" "ModuleUpdated" "ModuleLocked" "TimerFired" "DataEventChanged" "CCStatusChanged" "MainAudioStreamChanged" "NPTReferred" "MediaStarted" "MediaStopped" "MediaRepeated" "DataButtonPressed" "IPConnectionTerminated" "PeripheralEventOccured"), onoccur (Script), es_ref (URI), message_id (Number), message_version (Number), message_group_id (Number), module_ref (URI), language_tag (Number.), peripheral_ref (URI), time_mode ("absolute" "origAbsolute" "relativeToEvent" "relativeToLoad" "NPT"), time_value (CDATA), object_id (ID), subscribe (subscribe)	EMPTY
body&	invisible ("invisible")	n/a
div&	accesskey (Character), onfocus (Script), onblur (Script)	n/a
p&	accesskey (Character), onfocus (Script), onblur (Script)	n/a
span&	accesskey (Character), onfocus (Script), onblur (Script)	n/a
object&	remain ("remain"), accesskey (Character), streamstatus ("stop" "play" "pause"), streamposition (Number), streamlooping (Number), onfocus (Script), onblur (Script)	n/a

Table 5-2 BML module

Element	Attribute	Minimum content model
bml	I18N, version, xmlns	head, (body frameset)
bevent	id (ID)	(beitem)+
beitem	id (ID), type ("EventMessageFired" "EventFinished" "EventEndNotice" "Abort" "ModuleUpdated" "ModuleLocked" "TransmissionFinished" "TimerFired" "DataEventChanged" "CCStatusChanged" "MainAudioStreamChanged" "NPTReferred" "MediaStarted" "MediaStopped" "MediaRepeated" "DataButtonPressed" "IPConnectionTerminated" "PeripheralEventOccured"), "StoreFinished" "DataEventChangedEx" "SegmentPlayEnded" "MetadataUpdated"), onoccur (Script), es_ref (URI), message_id (Number), message_version (Number), message_group_id (Number), module_ref (URI), language_tag (Number.), peripheral_ref (URI), register_id (Number), service_id (Number), event_id (Number), time_mode ("absolute" "origAbsolute" "relativeToEvent" "relativeToLoad" "NPT"), time_value (CDATA), object_id (ID), segment_id (ID), subscribe (subscribe)	EMPTY
body&	invisible ("invisible")	n/a
div&	accesskey (Character), onfocus (Script), onblur (Script)	n/a
p&	accesskey (Character), onfocus (Script), onblur (Script)	n/a
span&	accesskey (Character), onfocus (Script), onblur (Script)	n/a
bdo&	orientation ("horiz" "vert")	n/a
a&	effect ("cut" "dissolve" "wipe1" "wipe2" "wipe3" "wipe4" "goosewing1" "goosewing2" "goosewing3" "goosewing4" "roll1" "roll2" "roll3" "roll4" "slide-in1" "slide-in2" "slide- in3" "slide-in4" "slide-out1" "slide-out2" "slide-out3" "slide- out4" "separate-wipe1" "separate-wipe2" "separate-wipe3" "separate-wipe4" "square-wipe1" "square-wipe2")	n/a

Element	Attribute	Minimum content model
object&	remain ("remain"), accesskey (Character), streamstatus ("stop" "play" "pause"), streamposition (Number), streamlooping (Number), streamspeednumerator (Number), streamstartposition (Number), streamendposition (Number), streamspeeddenominator (Number), streamlevel (Number), onfocus (Script), onblur (Script)	n/a

Table 5-3 Basic Mobile BML module

Element	Attribute	Minimum content model
bevent	id (ID)	(beitem)+
beitem	id (ID), type ("EventMessageFired" "ModuleUpdated" "ModuleLocked" "TimerFired" "DataEventChanged" "MainAudioStreamChanged" "MediaStopped"), onoccur (Script), es_ref (URI), message_id (Number), message_version (Number), message_group_id (Number), module_ref (URI), time_mode ("absolute" "origAbsolute"), time_value (CDATA), object_id (ID), subscribe (subscribe)	EMPTY
object&	accesskey (Character), streamstatus ("stop" "play" "pause"), onfocus (Script), onblur (Script)	n/a

Table 5-4 Server-based BML module

Element	Attribute	Minimum content model
bevent	id (ID)	(beitem)+
beitem	id (ID), type ("EventMessageFired" "ModuleUpdated" "ModuleLocked" "TimerFired" "DataEventChanged" "CCStatusChanged" "MainAudioStreamChanged" "NPTReferred" "MediaStopped" "DataButtonPressed" "IPConnectionTerminated" "StoreFinished" "DataEventChangedEx" "SegmentPlayEnded" "MetadataUpdated"), onoccur (Script), es_ref (URI), message_id (Number), message_version (Number), message_group_id (Number), module_ref (URI), language_tag (Number), time_mode ("absolute" "origAbsolute" "NPT"), time_value (CDATA), object_id (ID), segment_id (ID), subscribe (subscribe)	EMPTY
body&	invisible ("invisible")	n/a
div&	accesskey (Character), onfocus (Script), onblur (Script)	n/a
p&	accesskey (Character), onfocus (Script), onblur (Script)	n/a
span&	accesskey (Character), onfocus (Script), onblur (Script)	n/a
object&	remain ("remain"), accesskey (Character), streamstatus ("stop" "play" "pause"), streamposition (Number), streamlooping (Number),	n/a

	onfocus (Script), onblur (Script)	
--	--------------------------------------	--

5.3.20.1 Extensions for handling broadcasting service events

- bevent element

This element describes events to be processed in the document. The events described inside this element are events defined in the extension to the BML specifications. They must not include events for Intrinsic Event module. This element is described as a child element of a head element. It contains one or more beitem elements as its child elements.

- beitem element

This element describes mapping between an individual event and a procedural description. The attributes of beitem element are defined as follows:

- id attribute

The identifier of this beitem element.

- type attribute

This attribute specifies the type of an event. Table 5-5 shows the available values representing events that can be contained in a type attribute.

Table 5-5 Values Applicable to type Attribute

type Attribute values	Semantics
EventMessageFired	The event that notifies occurrence of a message from the broadcast station. When the message is transmitted as a general event message, this event occurs with a timing specified by the general event message descriptor (instant trigger, absolute time, relative time from the beginning of a program, NPT).
EventFinished	The event that notifies the end of a program
EventEndNotice	The event that prenotifies the end of a program
Abort	The event that notifies the abort of contents playback.
ModuleUpdated	The event that notifies the detection of an updated version of carousel module.
ModuleLocked	The event that notifies locking of the module specified by running LockModuleOnMemory() or lockModuleOnMemoryEx().
TransmissionFinished	The event that notifies the end of communication after execution of delayed calling registered by registerTransmission().
TimerFired	The event that notifies the trigger of a timer that starts at a specified time.
DataEventChanged	The event that notifies an update of data_event_id of a component that contains the BML document currently played.
CCStatusChanged	The event that notifies the switching of subtitle language and display state.
MainAudioStreamChanged	The event that defines the switching of main audio that is defaulted to -1 indicating component_tag.
NPTReferred	The event that notifies that GetNPT() and related function have

type Attribute values	Semantics
	been enabled.
MediaStopped	The event that notifies the end of decoding of the monomedia decoder.
MediaStarted	The event that notifies the beginning of decoding of the monomedia decoder.
MediaRepeated	The event that notifies the restart of decoding of the monomedia decoder.
DataButtonPressed	The event that notifies that the button which issues a command to switch to data broadcasting (specified in the operational guideline) has been pressed.

- onoccur attribute

This attribute specifies character strings describing a procedure that is executed when an event has occurred.

- es_ref attribute

This attribute specifies a URI that identifies an elementary stream (ES). An audio ES must contain a channel identification.

- message_id attribute

This attribute identifies each of the event messages distributed in the general event message descriptor.

It corresponds to the upper eight bits of event_msg_id (message identification), which is encoded in the general event message descriptor that is defined in ARIB STD-B24 Volume 3, Chapter 7.

- message_version attribute

This attribute specifies the version number of each event message identified by message_id attribute.

It corresponds to the lower eight bits of event_msg_id (message identification), which is encoded in the general event message descriptor that is defined in ARIB STD-B24 Volume 3, Chapter 7.

- message_group_id attribute

This attribute defines the value of event_msg_group_id (message group identification), which is encoded in the general event message descriptor that is defined in ARIB STD-B24 Volume 3, Chapter 7.

- module_ref attribute

This attribute defines a URI that identifies the module.

- language_tag attribute

When the type attribute is set to CCStatusChanged, language_tag attribute represents the subtitle language identification.

- register_id attribute

When the type attribute is set to TransmissionFinished, register_id attribute represents the registration ID that is returned from the registerTransmission() registration function for a delayed call.

- service_id attribute

This attribute contains service_id that identifies the service.

- event_id attribute

This attribute contains event_id that identifies an event.

- Peripheral_ref attribute

This attribute contains a URI string that identifying a external device.

- time_mode attribute

This attribute indicates the type of time specification. Table 5-4 shows the possible values of time_mode attribute.

Table 5-6 Values Applicable to time_mode Attribute

Values	Semantics
absolute	Absolute time during playback (Note 1)
origAbsolute	Reception time (Note 2)
relativeToEvent	Relative time from the beginning of the program
relativeToLoad	Relative time from the beginning of loading the BML document.
NPT	NPT time

Note 1: To play a content that is received in real time or stored in a storage device, the current time when the content is played is used as a base time. For example, to play a content that is received in real time, a time contained in TOT/TDT or a time calculated based on TOT/TDT may be referenced. To play a content in a storage device, a clock indicating the absolute time when the content is played may be referenced.

Note 2: To play a content that is received in real time or stored in a storage device, the time when the content is or was received is used as a base time. For example, to play a content that is received in real time, a clock using the contained in TOT/TDT or a time calculated based on TOT/TDT as a base time may be referenced. To play a content in a storage device, a clock using the PartialTS Time descriptor of SIT as a base time may be referenced.

- time_value attribute

This attribute indicates the time as numeric character strings. Table 5-5 shows the possible values of time_value attribute for the value of time_mode attribute.

Table 5-7 Values Applicable to time_value Attribute

time_mode attribute values	Semantics
absolute	Yyyymmddhhmmss
origAbsolute	Specified in a series of year (4 digits), month (2 digits), day (2 digits), hour (2 digits), minute (2 digits) and second (2 digits)
relativeToEvent	In milliseconds (decimal)
relativeToLoad	In milliseconds (decimal)
NPT	In milliseconds (decimal) (Note 1)

Note 1: Since NPT is a relative time to STC (33bit) which is based on 90KHz unit, this attribute must be set to the value obtained by dividing the actual NPT by 90.

- object_id attribute

This attribute specifies the id attribute of the object element.

- segment_id attribute

This attribute specifies a segment ID that identifies a video scene of server-based content.

- subscribe attribute

The only available value of this attribute is "subscribe." When a value other than "subscribe" or no value is set, the occurrence of this event must be ignored.

Table 5-6 shows the possible attributes for each type attribute. Since the id, onoccur and subscribe attributes are effective for all type attributes, they are not listed in this table.

Table 5-8 Relationship between type and Other Attributes

type Attribute	Possible Attributes	Semantics
EventMessageFired	es_ref message_group_id message_id message_version	<ul style="list-style-type: none"> - If subscribe attribute is specified, it must receive the event message section identified by es_ref, message_group_id, message_id, and message_version attributes. If no subscribe attribute is specified, it does not receive the event message. - If the event message identified by es_ref, message_group_id, message_id, and event_version attributes is received, it must execute the procedural description indicated by the onoccur attribute at the specified timing according to the descriptor of the general event message. - If no message_id attribute is specified or its value exceeds 255, an event message with any message_id must be processed. - If no message_version attribute is specified or its value exceeds 255, an event message with any message_version must be processed. - If subscribe attribute is set, it must continue receiving the event message. However, if event messages with the same message_id are received several times, it must execute the procedural description indicated by onoccur attribute only once when the event message having a different message_version from the last one is received. - If the es_ref attribute is not specified, the component carrying this BML document must be specified.
EventFinished		If subscribe attribute is specified, it must execute the procedural description indicated by the onoccur attribute when the notification of the end of the program currently presented is detected.
EventEndNotice		If subscribe attribute is specified, it must execute the procedural description indicated by the onoccur attribute when the previous notification of the end of the program currently presented is detected.
Abort		If subscribe attribute is specified, it must execute the procedural description indicated by the onoccur attribute when the notification of the abort of the contents representation is detected.
ModuleUpdated	module_ref	<p>If subscribe attribute is set, it must start receiving the module of the carousel specified by the module_ref attribute and any of its updated version. If no subscribe attribute is specified, it must not receive the module.</p> <p>This event must be occurred in following case:</p> <ol style="list-style-type: none"> 1) The receiver detected a transmission of this module when the receiving has just started, or new transmission of this module when this

type Attribute	Possible Attributes	Semantics
		<p>module did not transmit yet.</p> <p>2) The receiver detected no transmission of this module when the receiving has just started, or the transmission pause of this module while this module is being transmitted.</p> <p>3) After the process described in 1), the receiver detects the version update of this module. If update of the module that is identified by module_ref attribute is detected, the procedural description indicated by onoccur attribute is executed.</p>
ModuleLocked	module_ref	If subscribe attribute is set, it must execute the procedural description specified by onoccur attribute when the lock of the module identified by module_ref attribute is detected.
TransmissionFinished	register_id service_id event_id	<p>If subscribe attribute is set, it must execute the procedural description specified by onoccur attribute when the end of the delayed call is detected. The delayed call process is executed by the event specified by service_id and event_id attributes. It returns a value specified by register_id attribute.</p> <p>The register_id attribute must be specified.</p> <p>When the service_id is not specified, it is interpreted as that the service_id, which is presented currently, is designated.</p> <p>When the event_id is not specified, it is interpreted as that the event_id, which is presented currently, is designated.</p>
TimerFired	time_mode time_value	<p>If subscribe attribute is set, it must execute the procedural description indicated by onoccur attribute when a timer that was triggered at the time specified by the time_mode and time_value attributes. If the time specified by the time_value attribute when the timer is newly designated, e.g. at the time when a document is launched, when a subscribe attribute is set, and when a time_value attribute is set, it must execute the procedural description specified by onoccur attribute immediately.</p> <p>The time_mode attribute and the time_value attribute must be specified.</p>
DataEvent Changed		If subscribe attribute is set, it must execute the procedural description specified by onoccur attribute when update of data_event_id in the component which includes the currently presented BML document is detected.
CCStatusChanged	es_ref language_tag	<p>- If subscribe attribute is set, it must execute the procedural description indicated by onoccur attribute when the display state of language specified by language_tag attribute in the subtitle stream specified by es_ref attribute.</p> <p>- If no es_ref attribute is set, it must execute the</p>

type Attribute	Possible Attributes	Semantics
		<p>procedural description for any subtitle (except superimpose) stream.</p> <ul style="list-style-type: none"> - If no language_tag attribute is set, it must execute the procedural description for any language.
MainAudioStreamChanged	es_ref	<ul style="list-style-type: none"> - If subscribe attribute is set, it must execute the procedural description specified by onoccur attribute when the selection of voice stream in the main voice specified by es_ref attribute is changed. This event must not occur at the time when audio stream is selected immediately after the selection of the program channel - If no es_ref is set, it must execute the procedural description for the main voice stream or all channels in it.
NPTReferred	es_ref	<p>If the subscribe attribute is set, it must execute the procedural description indicated by the onoccur attribute only once when the functions that use NPT (e.g. getNPT() etc.) become available by receiving the NPT reference descriptor in the stream designated by the es_ref attribute. If the es_ref attribute is not specified, the component carrying this BML document must be specified.</p>
MediaStopped	object_id	<p>If the subscribe attribute is set, it must execute the procedural descriptor indicated by onoccur attribute when the presentation of monomedia specified by the object element with the value of the object_id attribute being id attribute has ended.</p> <p>The object_id attribute must be specified.</p>
MediaStarted	object_id	<p>If the subscribe attribute is set, it must execute the procedural descriptor indicated by onoccur attribute when the presentation of monomedia specified by object element with the value of object_id attribute being id attribute has started.</p> <p>The object_id attribute must be specified.</p>
MediaRepeated	object_id	<p>If the subscribe attribute is set, it must execute the procedural descriptor indicated by onoccur attribute when the presentation of monomedia specified by object element with the value of object_id attribute being id attribute has restarted the playback.</p> <p>The object_id attribute must be specified.</p>
DataButtonPressed		<p>If the subscribe attribute is set, it must execute the procedural descriptor indicated by onoccur attribute when the button which issues a command to switch to data broadcasting (specified in the operational guideline) is pressed.</p>
StoreFinished	es_ref	<p>When the subscribe attribute is set, it must execute the procedural descriptor indicated by onoccur attribute once a storing process that has been triggered by a storage function finishes.</p>

type Attribute	Possible Attributes	Semantics
		<p>The es_ref attribute is required. It must not be omitted.</p> <p>The es_ref attribute must specify which ES is used for storing. When the subscribe attribute is not specified, even if the storing process finishes the procedural descriptor is not executed.</p>
DataEventChangedEx	es_ref	<p>When the subscribe attribute is set, it must execute the procedural descriptor indicated by onoccur attribute once an updated data_event_id of a component specified in the es_ref attribute is detected. When the subscribe attribute is not specified, even if such an update is detected the procedural descriptor is not executed.</p>
SegmentPlayEnded	segment_id	<p>When the subscribe attribute is set, it must execute the procedural descriptor indicated by onoccur attribute once a play of a resource specified in segmentation metadata identified in segment_id finishes. When the subscribe attribute is not specified, even if such a play finishes the procedural descriptor is not executed.</p> <p>The es_ref attribute is required. It must not be omitted.</p>
MetadataUpdated		<p>When the subscribe attribute is set, it must execute the procedural descriptor indicated by onoccur attribute once an updated metadata belonging to server-based content belonging to the presenting BML document is detected. When the subscribe attribute is not specified, even if such an update is detected the procedural descriptor is not executed.</p>

5.3.20.2 Extension for monomedia presentation

- Attributes for Controlling Continuation of Monomedia Presentation between Documents
 - remain attribute

This attribute is added to the definition of object element.

The only possible value of this value is “remain”. When this value is specified, the documents share the objects. During transition between documents, the presentation states of the shared objects are retained. The attribute values of the object element and the CSS properties in a document before transition are inherited to the object element after transition. This feature enables continuous playback of video and audio between documents. The same set of the id attributes of the object element for which this attribute has been specified must be applied to both a document before transition and a document after transition.

Two or more ESs share the remain attribute for video or audio, as long as the ESs belong to a common content group. When a monomedia has been stored in the content memory using lockModuleOnMemory(), the monomedia can only be shared by the documents belonging to the document groups to which the concerned module lock is applicable.

- Extended Attributes for Controlling Stream Playback

The following attributes are added to the definitions of the object element.

- streamstatus attribute

This attribute specifies the state of a stream. The possible values are Stop, Play, and Pause. Note that the value Stop for an audio stream means Mute instead of Stop.

- streamposition attribute

The attribute indicates a current time position during stream playback. While in Pause, a current value is hold. While in Stop, it is set to 0 (zero). While in Play, its value is represented with the unit of time defined for each media type.

- streamlooping attribute

This attribute specifies the number of looping of stream playback. Its value is an integer with the initial value 1 (one).

- streamspeednumerator and streamspeeddenominator attributes

These attributes specify the speed of stream playback. The stream must be played back with the speed obtained by multiplying its original speed by streamspeednumerator / streamspeeddenominator.

The initial values : streamspeednumerator = 1 and streamspeeddenominator = 1

When streamspeednumerator = 0, the playback must be stopped. However, streamstatus must not be changed no matter it has been set to play.

Example:

Stop: streamspeednumerator = 0

Regular playback: streamspeednumerator = 1 and streamspeeddenominator = 1

Reverse playback: streamspeednumerator = -1 and streamspeeddenominator = 1

- streamlevel attribute

This attribute is an integer from -1 to 100. The value -1 indicates muting, while the value 0 means the minimum audible volume. The value 100 is the volume set by the receiver. The initial value must be 100.

- streamstartposition, streamendposition attributes

The streamstartposition attribute specifies a start time position to start playing a stored stream. The streamendposition attribute indicates an end time position to quit playing a stored stream. Each value is represented in the unit of time that is defined for each media type in an operational standard regulation. The initial value for the streamstartposition attribute must be set to 0, while the initial value for the streamendposition attribute must be set to -1. The -1 initial value for the streamendposition means that the concerned stream is played to the end. When the streamlooping attribute is available, the specified section of the concerned stream is repeated as specified in the streamlooping attribute.

- Specification of Character Orientation

- orientation attribute

This attribute specifies the orientation of a character composition. This is an additional attribute to the common attribute declaration “%18n.attrib;” and the definition of the bdo element for internationalization. The possible values are “horiz” and “vert”. The “horiz” value indicates the horizontal orientation (from left to right). The “vert” value indicates the vertical (from top to bottom) orientation. The initial value must be horiz.

5.3.20.3 Extension for special effect during screen transition

- effect attribute

This attribute is added to the definition of the anchor element. The effect attribute specifies the special effect performed during screen transition. The meaning of each value is described in Section 7.1.4 of ARIB STD-B5.

[cut | dissolve | wipe1 | wipe2 | wipe3 | wipe 4 | goosewing1 | goosewing2 | goosewing3 | goosewing4 | roll1 | roll2 | roll3 | roll4 | slide-in1 | slide-in2 | slide-in3 | slide-in4 | slide-out1 | slide-out2 | slide-out3 | slide-out4 | separate-wipe1 | separate-wipe2 | separate-wipe3 | separate-wipe4 | square-wipe1 | square-wipe2]

5.3.20.4 Extension for switching on and off display of whole document

- invisible attribute

This attribute is added to the definition of the body element. The available value to this attribute is limited to invisible. When the invisible attribute is specified, no elements and background of a BML document must be displayed irrespective to the specification of CSS characteristic of each element.

When the concerned service has video and audio streams selected in EPG or other, these streams must be played.

This attribute controls only the display state of a document. Even if this attribute is specified, an event occurs and a procedural descriptions is executed.

5.3.20.5 Extension for handling events

The onfocus attribute, the onblur attribute, and the accesskey attribute are added to the div element, the p element, the span element, and the object element.

5.3.20.6 Extension for describing root attribute for a BML document type

- bml attribute

root element: This element is used for structuring a whole BML document.

5.4 CSS-based Style Sheet

5.4.1 Media type

5.4.1.1 Definition

A style sheet specifies how document data is represented and displayed with various media (e.g. display, voice recognition device). With the assumption that certain properties of CSS are specific to each medium, there is a framework for applying each style sheet to a specific medium.

For this reason, each style sheet defined in this specification specifies its target medium, which is called media type.

5.4.1.2 Specification of media type

This Standard uses the @media rule for specifying a media type in style sheets. This rule can specify a target medium and the conventions applied to the medium. This Standard requires tv (lowercase) for

media type. The tv media type assumes a device like a television set that has color display capability and limited scroll functions.

5.4.1.3 Media group

The available media groups to the 'tv' media type are defined in Table 5-7. The value of each media group conforms to Section 7.3, CSS2

Table 5-9 Media Group Values for tv Media Type

Media Group	continuous/paged	visual/aural/tactile	grid/bitmap	interactive/static
Value	Both	Visual, aural	bitmap	both

5.4.2 Box model

5.4.2.1 Dimension of box

A box conforms to CSS2 8.1. A box has an area that contains text or images. A box has padding areas, border areas, and margin areas, as needed. The relationship among these areas follows CSS2 8.1.

5.4.2.2 Margin property

This property conforms to CSS2 8.3. It is used to specify the width of a margin area of a box

5.4.2.3 Padding property

This property conforms to CSS2 8.4. It is used to specify the width of a padding area of a box.

5.4.2.4 Border property

This property conforms to CSS2 8.5. It is used to specify the width, the color, and the style of a border area of a box. This specification requires the following additional border properties:

- border-top-color, border-right-color, border-left-color, border-bottom-color

The four borders must be rendered in the order of left, right, top, and bottom. Each of the four corners is overwritten according to this rendering order. When border-top-color, border-right-color, border-left-color, nor border-bottom-color is not specified, the initial value (white), instead of the value in the color property of the element, is assumed.

Value: <color> transparent | inherit

Initial value: white

Inheritance: No

Note: The specification accepts 'transparent' as a value available to the four properties. Although the CSS does not express 'transparent' as a valid value, the Errata in REC-CSS2-19980512 (<http://www.w3.org/Style/css2-updates/REC-CSS2-19980512-errata.html>) reads, "Section 8.5.2 Border color: 'border-top-color', 'border-right-color', 'border-bottom-color', 'border-left-color',

and 'border-color'[2001-06-25]

The value 'transparent' is also allowed on 'border-top-color', 'border-right-color', etc. Change the line "Value: <color> | inherit" to

Value: <color> | transparent | inherit."

Although the specification (ARIB STD-B24) has no reference to the correction, which accepts 'transparent', considering the correction and the requirements, has made 'transparent' a value available to the four properties in the specification.

- border-color-index property

This property is used to specify the colors of the four box borders using index colors, as defined in Section 5.4.6.1. Up to four values can be specified. When four values are specified, they are assigned to the top, right, bottom, and left borders in the specified order. When three values are specified, they are assigned to the top, right/left, and bottom borders in the specified order. When two values are specified, they are assigned to the top/bottom and right/left borders in the specified order. And when one value is specified, it is assigned to the four borders.

Value: <color-index>{1,4} | inherit

Initial value: 0 0 0 0

Applies to: All elements

Inheritance: No

Percentages: Not applied

Applicable media type: tv

- border-top-color-index property

This property specifies the top border color of a box with an index color.

Value: <color-index> | inherit

Initial value: 0

Applies to: All elements

Inheritance: No

Percentages: Not applied

Applicable media type: tv

- border-right-color-index property

This property specifies the right border color of a box with an index color.

Value: <color-index> | inherit

Initial value: 0

Applies to: All elements

Inheritance: No

Percentages: Not applied

Applicable media type: tv

- border-bottom-color-index property

This property specifies the bottom border color of a box with an index color.

Value: <color-index> | inherit

Initial value: 0

Applies to: All elements

Inheritance:	No
Percentages:	Not applied
Applicable media type:	tv

- border-left-color-index property

This property specifies the left border color of a box with an index color.

Value:	<color-index> inherit
Initial value:	0
Applies to:	All elements
Inheritance:	No
Percentages:	Not applied
Applicable media type:	tv

5.4.3 Visual formatting model

A visual formatting model specifies the model used by a browser for displaying a BML document.

5.4.3.1 Viewport

A viewport specifies an area on screen for presenting a BML document to a user. It conforms to CSS2 9.1.1.

5.4.3.2 Containing block

A containing block is a rectangular area based on that the location and size of a box is determined, as defined in CSS2. It conforms to CSS2 9.1.2.

5.4.3.3 Location specifications

There are three methods to specify the location of a box in CSS2: normal flow, float, and absolute location. These methods conform to CSS2 9.4, 9.5, and 9.6 respectively.

5.4.3.4 Layer format display

This Standard includes a three-dimensional location specification, or the specification on the z-axis, as well as the x-axis and the y-axis that is supported in CSS2. The three-dimensional location specification enables overlapped boxes to be displayed. The current specification supports CSS2 9.9.

5.4.4 Other visual effects

5.4.4.1 Display/Hide

This visual effect specifies whether to display or hide the box generated by an element. It conforms to CSS2 11.2.

5.4.5 Paged media

The paged media allows for a presentation in which data content is described on separate pages. It is supported in CSS2 under the @page rule.

5.4.6 Colors and backgrounds

The colors and backgrounds conform to CSS2 14.

5.4.6.1 Color unit

Each property in the style sheet accepts the following color units: keyword, RGB, YCBCR and index color.

- Specify by Keywords

The following 16 color keywords can be assigned to a color unit:

aqua, black, blue, fuchsia, grey, green, lime, maroon, navy, olive, purple, red, silver, teal, white, yellow

- Specify by RGB

This specification style uses one of the four numeric formats defined by Table 5-8.

Table 5-10 RGB Specification style

Format	Meaning
#rgb	RGB values are specified in hexadecimal format. Conversion to six-digit notation is done by duplicating each digit: #96e is converted to #9966ee.
#rrggbb	RGB values are written in hexadecimal format with range 0 to 255.
rgb(0-255,0-255,0-255)	RGB values are written in decimal format with range 0 to 255.
rgb(0.0%-100.0%,0.0%-100.0%,0.0%-100.0%)	RGB values are written in percents format with range 0.0% to 100.0%.

- Specify by YCBCR

This specification style uses one of the two numeric formats defined by Table 5-9.

Table 5-11 YCBCR Specification style

Format	Meaning
YCBCR(0-255,0-255,0-255,0-255)	Y, CB, CR, and Alpha values are written in decimal integers with range 0 to 255.
YCBCR(0.0%-100.0%,0.0%-100.0%,0.0%-100.0%,0.0%-100.0%)	Y, CB, CR, and Alpha values are written in percents with range 0.0% to 100.0%.

- Index Color

An index color is specified using an index number <color-index> of the color lookup table (CLUT). The index number is an integer following 0.

The following extended properties are used to set CLUT.

- clut property

This property specifies the resource of color map data to be set to the CLUT using URI. The color map data conforms to the encoding scheme defined in Annex A. If no clut property is specified explicitly, the receiver's default CLUT settings, which is defined in an operational standard regulation, are retained.

Value:	<uri>
Initial value:	Arbitrary
Applies to:	All elements
Inheritance:	Yes
Percentages:	Not applicable
Applicable media type:	tv

5.4.6.2 Color specification

This property conforms to CSS2 14.1. However, for color properties, the foreground color of an element must be a color specified with a method other than the index color method. The following property is added:

- color-index property

This property specifies the foreground color of an element with an index color.

Value:	<color-index> inherit
Initial value:	0
Applies to:	All elements
Inheritance:	Yes
Percentages:	Not applicable
Applicable media type:	tv

5.4.6.3 Background

Each back ground property conforms to CSS2 14.2. However, for background-color properties, the background color must be a color specified with a method other than the index color method. The following property is added:

- background-color-index property

This property specifies the background color of an element with an index color.

Value:	<color-index> inherit
Initial value:	0
Applies to:	All elements
Inheritance:	No
Percentages:	Not applicable
Applicable media type:	tv

5.4.7 Fonts

The font properties and general rules conform to CSS2 15.

5.4.7.1 Font equipment

For European language fonts, general fonts (serif, sans-serif, cursive, fantasy, and monospace) are allocated. Font types required to support the Japanese language are specified separately in an operational standard regulation.

5.4.7.2 Font property

Each font property conforms to CSS2 15.2.

5.4.8 Text

Each text property conforms to CSS2 16. However, the word-spacing property is applied only to spacing behaviour between words in any of the European language character sets. The text-transform property is not applied to the Japanese language.

5.4.9 Pseudo classes and pseudo elements

Each pseudo class conforms to CSS2 5.11.

5.4.10 Table related properties

Each property conforms to CSS2 17.

5.4.11 User interface

5.4.11.1 Frame display

The frame display function of CSS is different from the border properties of a block element, as specified in this standard, in that:

- No layout space is required for displaying the frames themselves.
- The frame shape is not necessarily a square, and it depends on the shape of a block element.

Each property conforms to CSS2 18.4. However, for outline-color properties, the frame color must be specified with a method other than the index color method. The following property is added:

- outline-color-index property

This property specifies the frame color with an index color.

Value:	<color-index> inherit
Initial value:	0
Applies to:	All elements
Inheritance:	No
Percentages:	Not applicable
Applicable media type:	tv

5.4.11.2 Cursor

This property conforms to CSS2 18.1.

5.4.12 Aural style sheet

This property conforms to CSS2 19.

5.4.13 Extended properties

The following properties are the extended properties to support the tv media type defined in this standard.

5.4.13.1 Properties for specifying screen size

- resolution property

This property specifies the resolution of a text graphic plane.

Value:	1920x1080 1280x720 960x540 720x480
Initial value:	960x540
Applies to:	body element
Inheritance:	No
Percentages:	Not applied
Applicable media type:	tv

- display-aspect-ratio property

This property specifies the displayed aspect ratio of a text graphic plane.

Value:	4v3 16v9
Initial value:	16v9
Applies to:	body element
Inheritance:	No
Percentages:	Not applied
Applicable media type:	tv

5.4.13.2 Properties for drawing with greyscale

- greyscale-color-index property

This property specifies greyscale colors with index colors. The leftmost property indicates a color nearest to the foreground color. For example, in the case of four-step greyscale, two colors (except foreground and background) are specified using this property.

Value:	<color-index>+ inherit
Initial value:	Specified by the broadcaster
Applies to:	All elements

Inheritance:	Yes
Percentages:	Not applied
Applicable media type:	tv

5.4.13.3 Properties for describing remote control operations

The following properties provide the extension necessary to describe focus movement operations with a remote control.

- nav-index property

This property specifies an index for an element to which the focus is applied. Any element with this property set to none, is given no focus. The focus is initially applied to the element with this property set to 0. Any value of this property must be unique in a BML document.

Value:	<integer> none
Initial value:	none
Applies to:	All elements where the focus can be set
Inheritance:	No
Percentages:	Not applied
Applicable media type:	tv

- nav-up property

This property specifies the value of the nav-index property describing an element to which the focus is applied when the up arrow key (↑) is pressed.

Value:	<integer> none
Initial value:	none
Applies to:	All elements where the focus can be set
Inheritance:	No
Percentages:	Not applied
Applicable media type:	tv

- nav-down property

This property specifies the value of the nav-index property describing an element to which the focus is applied when the down arrow key (↓) is pressed.

Value:	<integer> none
Initial value:	none
Applies to:	All elements where the focus can be set
Inheritance:	No
Percentages:	Not applied
Applicable media type:	tv

- nav-right property

This property specifies the value of the nav-index property describing an element to which the focus is applied when the right arrow key (→) is pressed.

Value:	<integer> none
Initial value:	none
Applies to:	All elements where the focus can be set
Inheritance:	No
Percentages:	Not applied
Applicable media type:	tv

- nav-left property

This property specifies the value of the nav-index property describing an element to which the focus is applied when the left arrow key (←) is pressed.

Value:	<integer> none
Initial value:	none
Applies to:	All elements where the focus can be set
Inheritance:	No
Percentages:	Not applied
Applicable media type:	tv

Focus movements using a combination of the above properties are defines as follows:

1. When none is specified to one or more of the nav-left, nav-right, nav-up, and nav-down properties, the focus does not move in the specified direction(s) with a remote control.
2. When the focus is applied to an element with only nav-index property specified (i.e. the nav-left, nav-right, nav-up, and nav-down properties are not specified), the focus stays in the element.
3. When both the tabindex and nav-index properties are specified, the nav-index property overrides the tabindex property. If either of these properties is not specified, it must be recognized that the tabindex property is implicitly specified, as defined in XHTML1.0.
4. When the focus is applied to an element with the visibility property set to hidden by pressing an arrow key whose direction is applicable to give the focus to the hidden element, the applied focus is enabled. In other words, the focus skips that element. If there is no corresponding specification of focus movement, the focus does not move.

5.4.13.4 Properties for exclusive control with remote control keys

- used-key-list property

This property specifies the type of remote control keys to be accepted by a BML browser.

Value:	<key-group>+ none
Initial value:	basic data-button
Applies to:	body element
Inheritance:	No
Percentages:	Not applied
Applicable media type:	tv

Table 5-10 lists the acceptable <key-group> character strings and their meaning.

Table 5-12 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 operations for data broadcasting (e.g. Red, Green, Blue, and Yellow color 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 media.

Note 4: Any receiver must provide the Power key. Masking power key by the BML contents is not allowed.

1. When a remote control key that is included in <key-group> specified in the used-key-list property is pressed, it must be processed as a key entry to the BML browser.
2. When a remote control key that is not specified in the used-key-list is pressed, it must be handled as a receiver function key (e.g. channel keys) and no event for the BML browser occurs.
3. When used-key-list is set to “none,” no remote key operation generates an event for the BML browser.

Example: When used-key-list is set to “basic,” a numeric key entry is interpreted as a channel selection with no effects to the operation of the BML contents. However, when used-key-list is set to “numeric-tuning” and the input element has the focus, a numeric key entry is interpreted as a numeric entry to the input element, not as a channel selection.

5.4.13.5 Extension properties used for services for portable/mobile receiving system

The following properties defined in Chapters 17 through 19 (pages 41 through 60), WAP CSS Specification are assumed to be used as extension properties used for services for portable/mobile receiving systems.

- -wap-marquee
- -wap-marquee-style
- -wap-marquee-loop
- -wap-marquee-dir
- -wap-marquee-speed
- -wap-accesskey
- -wap-input-format
- -wap-input-required

Chapter 6 Converting XML Document into BML Using XSL

6.1 Structure of XSL Documents

The structure of any XSL document must conform to the XML document conventions defined by B-XML.

For the purpose of transformation, an XSL document uses the language specifications for transforming XML documents described in the XSL standard developed by W3C- and do not use the formatting specifications.

Example: The following list shows an example of the XSL description.

<code><xsl:stylesheet</code>	Beginning of style sheet tag
<code><xsl:template match="mytag"></code>	Beginning of template
<code>Xxx</code>	BML and other tags
<code><xsl:apply-templates/></code>	Apply template
<code>Yyy</code>	BML and other tags
<code></xsl:template></code>	End of template
<code>...</code>	
<code></xsl:stylesheet></code>	End of style sheet tag

6.2 XSLT Specifications

This section explains the XSL Transformation (XSLT) specifications that are used for transforming XSL and XML documents into BML documents. XSLT is based on the specifications defined in Reference (4). This section specifically describes items to be defined to implement the B-XML structure.

6.2.1 XSLT

XSLT defined in B-XML1.0 is a language for transforming documents described in XML application language into BML documents.

6.2.2 Character encoding

In XSLT, the character encoding for character strings in String type functions and elements, and the character coding for the encoding declaration (`<?xml encoding="..."?>`) conforms to BML and the B-XML specifications.

6.2.3 Numbers

Handling numbers in XSLT follows the XSLT specifications. However, it is allowed to limit the size of Number as less than 64 bits.

6.2.4 XSLT style sheet elements

Table 6-1 lists the types of elements in XSLT style sheet.

Table 6-1 XSLT Tags

Tag Name	Purpose
<xsl:stylesheet>	Declares a style sheet.
<xsl:transform>	Same as <xsl:stylesheet>.
<xsl:import>	Imports a style sheet.
<xsl:include>	Includes another style sheet.
<xsl:strip-space>	Deletes an element type from the set of white-space-preserving elements.
<xsl:preserve-space>	Adds an element types to the set of white-space-preserving elements.
<xsl:key>	Declares a key.
<xsl: decimal-format>	Declares locale which controls the interpretation of the format pattern.
<xsl:attribute-set>	Defines a set of named attributes.
<xsl:variable>	Specifies a value to a variable.
< xsl:param>	Specifies a parameter. Obtains a parameter from a source other than the style sheet.
<xsl:template>	Specifies a template convention.
<xsl:apply-templates>	In the absence of select attribute, processes all of children (including text nodes) of the current node. If select attribute is set, processes the node selected by the expression.
<xsl:call-template>	Invokes a template specified by name attribute.
<xsl:apply-imports>	Processes the current node using only the template rules (including the current rule) that has been imported into the style sheet.
<xsl:for-each>	Executes a loop.
<xsl:value-of>	Replaced by the result of evaluation of an expression specified by select attribute.
<xsl:copy-of>	Copies a node list specified by select attribute.
<xsl:number>	Inserts formatted number into the result tree.
<xsl:choose>	Select one of the possible selection items.
<xsl:if>	Provides an if-then condition.
<xsl:text>	Describes character string data.
<xsl:copy>	Provides an easy way to copy the current node.
<xsl:message>	Sends a message in a way that is dependent on the XSLT processor.
<xsl:processing-instruction>	Instantiated to generate a processing instruction node.
<xsl:comment>	Instantiated to generate a comment node in the result tree.
<xsl:element>	Generates an element with a name specified by name attribute (required) and namespace attribute (optional).
<xsl:attribute>	Used to add an attribute to result elements whether created by literal result element in the style sheet or by instructions such as xsl:element.
<xsl:when>	Instantiates the contents of an element according to the result of a conditional branch with test attribute.
<xsl:otherwise>	If <xsl:when> is not true, instantiates the contents of an element.
<xsl:sort>	Executes sort.
< xsl:with-param >	Specifies a parameter.

Tag Name	Purpose
<xsl:output>	Specifies a method of outputting the result tree.
<xsl:fallback>	Specifies a template that is used when no extensional element is available.

Chapter 7 Procedural Description Language

7.1 DOM API

This section defines DOM (Document Object Model) APIs for a BML document. The APIs conform to W3C Recommendation DOM Level 1.

7.1.1 Core DOM Fundamental Interfaces

Core DOM Fundamental Interfaces are the primary interfaces in Core DOM Interfaces. These interfaces must be implemented.

The interfaces, methods, and attributes included in Core DOM Fundamental Interfaces follow Fundamental Interfaces of DOM Level 1, Section 1.2.

7.1.2 Core DOM Extended interfaces

Core DOM Extended Interfaces are extensional part of Core DOM Interfaces.

The interfaces, methods, and attributes included in Core DOM Extended Interfaces follow Extended Interfaces of DOM Level 1, Section 1.3.

7.1.3 HTML DOM interfaces

HTML DOM Interfaces are DOM Interfaces for operating specific elements and attributes defined in HTML.

As stated in Section 5.3, BML elements are based on Modularization of XHTML that defines Strict document type of XHTML 1.0 and its modularization. So these Interfaces are applicable to BML documents.

The methods and attributes of this interface conform to the conventions in DOM Level 1 Chapter 2, Document Object Model (HTML) Level 1. However, the following interfaces are not defined because they correspond to the elements that are not defined in the modules included in the Strict document type of Modularization of XHTML and Frames module.

- HTMLIsIndexElement interface
- HTMLDirectoryElement interface
- HTMLMenuElement interface
- HTMLBaseFontElement interface
- HTMLFontElement interface
- HTMLAppletElement interface

The following attributes are not defined because they correspond to the modules that are not defined in the Strict document type of Modularization of XHTML and Frames module.

Attributes of HTMLPreElement interface	width
Attributes of HTMLHeadingElement interface	align
Attributes of HTMLHRElement interface	align, noshade, size, and width
Attributes of HTMLDivElement interface	align
Attributes of HTMLParagraphElement interface	align
Attributes of HTMLBRElement interface	clear

Attributes of HTMLBaseElement interface	target
Attributes of HTMLLinkElement interface	target
Attributes of HTMLDListElement interface	compact
Attributes of HTMLOLListElement interface	compact, type, and start
Attributes of HTMLULListElement interface	compact and type
Attributes of HTMLLIElement interface	type and value
Attribute of HTMLFormElement interface	target
Attribute of HTMLInputElement interface	align
Attribute of HTMLLegendElement interface	align
Attribute of HTMLTableCaptionElement interface	align
Attributes of HTMLTableElement interface	align and bgColor
Attributes of HTMLTableCellElement interface	bgColor, height, noWrap, and width
Attribute of HTMLTableRowElement interface	bgColor
Attributes of HTMLImageElement interface	align, border, hspace, and vspace
Attribute of HTMLAreaElement interface	target
Attributes of HTMLObjectElement interface	align, border, hspace, and vspace
Attribute of HTMLFrameElement interface	target
Attribute of HTMLIFrameElement interface	align
Attributes of HTMLBodyElement interface	aLink, background, bgColor, link, text, and vLink

7.1.4 CSS DOM interface

CSS DOM Interface is an extensional DOM interface for operating a CSS-base style sheet of BML.

- BMLCSS2Properties Interface

This interface is a DOM interface for operating CSS properties associated with BML elements.

Interface definition:

```

interface BMLCSS2Properties {
    // CSS2 Conformant Properties
    // Box Model
    attribute DOMString    marginTop;           // margin-top
    attribute DOMString    marginRight;        // margin-right
    attribute DOMString    marginBottom;       // margin-bottom
    attribute DOMString    marginLeft;         // margin-left
    attribute DOMString    margin;             // margin
    attribute DOMString    paddingTop;         // padding-top
    attribute DOMString    paddingRight;       // padding-right
    attribute DOMString    paddingBottom;     // padding-bottom

```

attribute DOMString	paddingLeft;	// padding-left
attribute DOMString	padding;	// padding
attribute DOMString	borderTopWidth;	// border-top-width
attribute DOMString	borderRightWidth;	// border-right-width
attribute DOMString	borderBottomWidth;	// border-bottom-width
attribute DOMString	borderLeftWidth;	// border-left-width
attribute DOMString	borderWidth;	// border-width
attribute DOMString	borderTopColor;	// border-top-color
attribute DOMString	borderRightColor;	// border-right-color
attribute DOMString	borderBottomColor;	// border-bottom-color
attribute DOMString	borderLeftColor;	// border-left-color
attribute DOMString	borderColor;	// border-color
attribute DOMString	borderTopStyle;	// border-top-style
attribute DOMString	borderRightStyle;	// border-right-style
attribute DOMString	borderBottomStyle;	// border-bottom-style
attribute DOMString	borderLeftStyle;	// border-left-style
attribute DOMString	borderStyle;	// border-style
attribute DOMString	borderTop;	// border-top
attribute DOMString	borderRight;	// border-right
attribute DOMString	borderBottom;	// border-bottom
attribute DOMString	borderLeft;	// border-left
attribute DOMString	border;	// border
// Visual Formatting Model		
attribute DOMString	position;	// position
attribute DOMString	left;	// left
attribute DOMString	top;	// top
attribute DOMString	width;	// width
attribute DOMString	height;	// height
attribute DOMString	zIndex;	// z-index
attribute DOMString	lineHeight;	// line-height
attribute DOMString	verticalAlign;	// vertical-align
attribute DOMString	display;	// display
attribute DOMString	bottom;	// bottom
attribute DOMString	right;	// right
attribute DOMString	cssFloat;	// float
attribute DOMString	clear;	// clear
attribute DOMString	direction;	// direction

attribute DOMString	unicodeBidi;	// unicode-bidi
attribute DOMString	minWidth;	// min-width
attribute DOMString	maxWidth;	// max-width
attribute DOMString	minHeight;	// min-height
attribute DOMString	maxHeight;	// max-height
// Visual Effects		
attribute DOMString	visibility;	// visibility
attribute DOMString	overflow;	// overflow
attribute DOMString	clip;	// clip
// Generated Content, Automatic Numbering, and Lists		
attribute DOMString	content;	// content
attribute DOMString	quotes;	// quotes
attribute DOMString	counterReset;	// counter-reset
attribute DOMString	counterIncrement;	// counter-increment
attribute DOMString	markerOffset;	// marker-offset
attribute DOMString	listStyleType;	// list-style-type
attribute DOMString	listStyleImage;	// list-style-image
attribute DOMString	listStylePosition;	// list-style-position
attribute DOMString	listStyle;	// list-style
// Paged Media		
attribute DOMString	size;	// size
attribute DOMString	marks;	// marks
attribute DOMString	pageBreakBefore;	// page-break-before
attribute DOMString	pageBreakAfter;	// page-break-after
attribute DOMString	pageBreakInside;	// page-break-inside
attribute DOMString	page;	// page
attribute DOMString	orphans;	// orphans
attribute DOMString	widows;	// widows
// Backgrounds		
attribute DOMString	background;	// background
attribute DOMString	backgroundColor;	// background-color
attribute DOMString	backgroundImage;	// background-image
attribute DOMString	backgroundRepeat;	// background-repeat
attribute DOMString	backgroundPosition;	// background-position
attribute DOMString	backgroundAttachment;	// background-attachment
// Fonts		
attribute DOMString	color;	// color

attribute DOMString	fontFamily;	// font-family
attribute DOMString	fontStyle;	// font-style
attribute DOMString	fontSize;	// font-size
attribute DOMString	fontVariant;	// font-variant
attribute DOMString	fontWeight;	// font-weight
attribute DOMString	font;	// font
attribute DOMString	fontStretch;	// font-stretch
attribute DOMString	fontSizeAdjust;	// font-size-adjust
// Text		
attribute DOMString	textIndent;	// text-indent
attribute DOMString	textAlign;	// text-align
attribute DOMString	textDecoration;	// text-decoration
attribute DOMString	textShadow;	// text-shadow
attribute DOMString	letterSpacing;	// letter-spacing
attribute DOMString	wordSpacing;	// word-spacing
attribute DOMString	textTransform;	// text-transform
attribute DOMString	whiteSpace;	// white-space
// Tables		
attribute DOMString	captionSide;	// caption-side
attribute DOMString	borderCollapse;	// border-collapse
attribute DOMString	borderSpacing;	// border-spacing
attribute DOMString	tableLayout;	// table-layout
attribute DOMString	emptyCells;	// empty-cells
attribute DOMString	speakHeader;	// speak-header
// User Interface		
attribute DOMString	outlineColor;	// outline-color
attribute DOMString	outlineWidth;	// outline-width
attribute DOMString	outlineStyle;	// outline-style
attribute DOMString	outline;	// outline
attribute DOMString	cursor;	// cursor
// Aural Style Sheets		
attribute DOMString	volume;	// volume
attribute DOMString	speak;	// speak
attribute DOMString	pauseBefore;	// pause-before
attribute DOMString	pauseAfter;	// pause-after
attribute DOMString	pause;	// pause
attribute DOMString	cueBefore;	// cue-before

attribute DOMString	cueAfter;	// cue-after
attribute DOMString	cue;	// cue
attribute DOMString	playDuring;	// play-during
attribute DOMString	azimuth;	// azimuth
attribute DOMString	elevation;	// elevation
attribute DOMString	speechRate;	// speech-rate
attribute DOMString	voiceFamily;	// voice-family
attribute DOMString	pitch;	// pitch
attribute DOMString	pitchRange;	// pitch-range
attribute DOMString	stress;	// stress
attribute DOMString	richness;	// richness
attribute DOMString	speakPunctuation;	// speak-punctuation
attribute DOMString	speakNumeral;	// speak-numeral
// BML Extended Properties		
attribute DOMString	clut;	// clut
attribute DOMString	colorIndex;	// color-index
attribute DOMString	backgroundColorIndex;	//background-color-index
attribute DOMString	borderColorIndex;	// border-color-index
attribute DOMString	borderTopColorIndex;	//border-top-color-index
attribute DOMString	borderRightColorIndex;	//border-right-color-index
attribute DOMString	borderBottomColorIndex;	//border-bottom-color-index
attribute DOMString	borderLeftColorIndex;	//border-left-color-index
attribute DOMString	outlineColorIndex;	// outline-color-index
attribute DOMString	resolution;	// resolution
attribute DOMString	displayAspectRatio;	// display-aspect-ratio
attribute DOMString	grayscaleColorIndex;	// greyscale-color-index
attribute DOMString	navIndex;	// nav-index
attribute DOMString	navUp;	// nav-up
attribute DOMString	navDown;	// nav-down
attribute DOMString	navLeft;	// nav-left
attribute DOMString	navRight;	// nav-right
attribute DOMString	usedKeyList;	// used-key-list
attribute DOMString	wapMarqueeStyle;	// wap-marquee-style
attribute DOMString	wapMarqueeLoop;	// wap-marquee-loop
attribute DOMString	wapMarqueeDir;	// wap-marquee-dir
attribute DOMString	wapMarqueeSpeed;	// wap-marquee-speed
attribute DOMString	wapAccesskey;	// wap-accesskey

```

    attribute DOMString      wapInputFormat;           // wap-input-format
    attribute DOMString      wapInputRequired;         // wap-input-required
};

```

Attributes:

The CSS property value for each attribute is retained. See the conventions for BML CSS properties in Section 5.4.

Method:

None

7.1.5 Event DOM interface

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 BML event.

- BMLEvent Interface

BMLEvent Interface retains the context information of a BML event.

Interface definition:

```

interface BMLEvent {
    readonly attribute DOMString      type;
    readonly attribute HTMLInputElement  target;
};

```

Attributes:

type	Name of event
target	Target of event

For example, an event for broadcasting service uses BMLBeitemElement Interface, which is a BML element DOM interface for beitem element.

Method:

None

- BMLIntrinsicEvent Interface

BMLIntrinsicEvent Interface retains the Intrinsic Event context information of a BML event. It is a BMLEvent with attributes specific to Intrinsic Event.

Interface definition:

```

interface BMLIntrinsicEvent : BMLEvent {
    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

- BMLBeventEvent Interface

BMLBeventEvent Interface retains the context information of some BML events, more specifically, broadcasting service events. This interface is a BMLEvent with attributes specific to broadcasting service events.

Interface Definition:

```
interface BMLBeventEvent : BMLEvent {
    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 unsigned short    registerId;
    readonly attribute DOMString         serviceId;
    readonly attribute DOMString         eventId;
    readonly attribute BMLObjectElement  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.

Table 7-1 lists the values of status for each event.

Table 7-1 Events and Corresponding status Values

type Attribute of beitem	Value of status
EventMessageFired	0 Event message is received. -1 Error occurred when receiving event message.
EventFinished	0 Program has finished. 
EventEndNotice	0 Prenotification of the end of program occurred.
Abort	0 Abort of contents playback occurred normally.
ModuleUpdated	2 Detected sending of the module when the receiver starts receiving, or detected beginning of sending the module when its sending has stopped. 1 Detected not going to send the module the receiver starts receiving or detected stop of sending the module when it has been sent. 0 Module update occurred after detecting the beginning of sending the module (status=2). -1 Error on module reception occurred
ModuleLocked	0 Detected the locking of module. -1 Error on locking of module occurred -2 Found no specified module, in the case of an event to

type Attribute of beitem	Value of status
	lockModuleOnMemoryEx() -3 Could not lock because of an insufficient cash capacity, in the case of an event to lockModuleOnMemoryEx()
TransmissionFinished	0 Delayed call process ended. -1 Error on delayed call execution occurred.
TimerFired	0 Timer has triggered. -1 Error on timer processing occurred
DataEventChanged	1 Empty carousel was detected when switching “data_event_id”. 0 “data_event_id” update occurred normally. -1 Error occurred.
DataButtonPressed	0 The button that issues a command to switch to data broadcasting (specified in the operational guideline) is pressed.
CCStatusChanged	1 Subtitle is in display state. 0 Subtitle is in hidden state. -1 Error on subtitle occurred.
MainAudioStreamChanged	1 Audio component channel is selected. 0 Audio component channel is deselected. -1 Error on voice component channel selection occurred.
NPTReferred	0 Timer processing and execution of GetNPT() with NPT specification have been enabled after the NPT reference descriptor was received. -1 Error on reception of NPT reference descriptor occurred.
MediaStopped	0 Detected end of presentation. -1 Error on end of presentation occurred.
MediaStarted	0 Detected beginning of presentation. -1 Error on beginning of presentation occurred.
MediaRepeated	0 Detected a head of repeated media. -1 Error on detecting a head of repeated media.
IPConnectionTerminated	0 An IP connection that had been established by the connectPPP() function or the connectPPPWithISPParams()function was terminated.
PeripheralEventOccurred	0 A request of a data transfer from a peripheral device was detected. -1 During a data transfer from a peripheral device, an error occurred.
StoreFinished	1 A storing process was interrupted. 0 A storing process successfully finished. -1 A storing process failed
DataEventChangedEx	1 An empty carousel was detected when data_event_id of the ES specified in the es_ref attribute was updated. 0 data_event_id of the ES specified in the es_ref attribute was updated. -1 An error occurred.
SegmentPlayEnded	0 A playback of one of secified segments finished. -1 During a playback of a segment, an error occurred.
MetadataUpdated	0 A metadata set of the server-based content containing the current BML document was updated.

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 event, 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.
registerId	If the event is TransmissionFinished, the registration ID given by registering the delayed call. For other events, 0 (zero).
serviceId	If the event is TransmissionFinished, service_id of the BML document that registered the delayed call. For other events, 0 (zero).
eventId	If the event is TransmissionFinished, the event_id of the BML document that registered the delayed call. For other events, 0 (zero).
peripheralRef	If the event is PeripheralEventOccuerred, peripheralRef is the URI character string that identifies the sending peripheral device. For other events, an empty string.
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

7.1.6 BML extended DOM interface

This interface is an extended DOM interface for operating elements and attributes defined in BML. It is HTML DOM Interface defined in Section 7.1.3 with an extension for BML.

7.1.6.1 BML Document DOM Interface

This interface operates the whole BML document.

BMLDocument Interface

This interface operates the whole BML document. It is HTMLDocument Interface defined in Section 7.1.3 with methods for obtaining context information of the event currently processed and methods for obtaining the BML object element that has the focus.

Interface Definition:

```
interface BMLDocument : HTMLDocument {  
    readonly attribute BMLEvent      currentEvent;  
    readonly attribute BMLElement    currentFocus;  
};
```

Attributes:

currentEvent	Context information that indicates the event currently processed.
currentFocus	BML object element that has the focus.

Method:

None

7.1.6.2 BML element DOM interface

This interface operates BML element attributes and CSS properties.

As stated in Section 5.3, BML 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 (1) extended HTML DOM interfaces for operating the CSS properties of BML elements, (2) interfaces for handling elements with attributes added for broadcasting service, and (3) interfaces for the new elements are required.

This section defines the above three interfaces, (1), (2), and (3).

(1) Interfaces with Extension for Operating CSS Properties

The following interfaces are HTML DOM interfaces defined in Section 7.1.3 with an extension required to operate the CSS property of a BML document.

As the following example shows, each interface inherits an HTML DOM interface that corresponds to the element with an extension of style, normalStyle, focusStyle, and activeStyle attributes that are BMLCSS2Properties objects for retaining the CSS properties. The name of each interface is that of an HTML DOM interface with “HTML” replaced with “BML.”

Example: BMLBlockquoteElement Interface

Interface Definition:

```
interface BMLBlockquoteElement : HTMLBlockquoteElement {  
    attribute BMLCSS2Properties style;  
    attribute BMLCSS2Properties normalStyle;  
    attribute BMLCSS2Properties focusStyle;  
    attribute BMLCSS2Properties activeStyle;  
};
```

Attributes:

style	Retains the CSS properties specified in the style attribute of the element.
normalStyle	Retains the inherited value of CSS property that is applied for presentation in normal state.

And 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 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 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.

Methods:

None

- BMLElement Interface

This interface is used for the following 24 elements. It is an extension to HTMLElement.

address, abbr, acronym, cite, code, dfn, em, kbd, samp, strong, var, b, big, i, small, sub, sup, tt, bdo, dd, dt, noscript, noframes, and head

- BMLBlockquoteElement Interface

This interface is used for the blockquote element. It is an extension to HTMLBlockquoteElement.

- BMLPreElement Interface

This interface is used for the pre element. It is an extension to HTMLPreElement.

- BMLHeadingElement Interface

This interface is used for the h1, h2, h3, h4, h5, and h6 elements. It is an extension to HTMLHeadingElement.

- BMLHRElement Interface

This interface is used for the hr element. It is an extension to HTMLHRElement.

- BMLQuoteElement Interface

This interface is used for the q element. It is an extension to HTMLQuoteElement.

- BMLBRElement Interface

This interface is used for the br element. It is an extension to HTMLBRElement.

- BMLModElement Interface

This interface is used for the ins and del elements. It is an extension to HTMLModElement.

- BMLLinkElement Interface

This interface is used for the link element. It is an extension to HTMLLinkElement.

- BMLDListElement Interface

This interface is used for the dl element. It is an extension to HTMLDListElement.

- BMLOLListElement Interface

This interface is used for the ol element. It is an extension to HTMLOLListElement.

- BMLUListElement Interface
This interface is used for the ul element. It is an extension to HTMLUListElement.
- BMLLIElement Interface
This interface is used for the li element. It is an extension to HTMLLIElement.
- BMLButtonElement Interface
This interface is used for the button element. It is an extension to HTMLButtonElement.
- BMLFieldSetElement Interface
This interface is used for the fieldset element. It is an extension to HTMLFieldSetElement.
- BMLInputElement Interface
This interface is used for the input element. It is an extension to HTMLInputElement.
- BMLLabelElement Interface
This interface is used for the label element. It is an extension to HTMLLabelElement.
- BMLLegendElement Interface
This interface is used for the legend element. It is an extension to HTMLLegendElement.
- BMLOptGroupElement Interface
This interface is used for the optgroup element. It is an extension to HTMLOptGroupElement.
- BMLOptionElement Interface
This interface is used for the option element. It is an extension to HTMLOptionElement.
- BMLSelectElement Interface
This interface is used for the select element. It is an extension to HTMLSelectElement.
- BMLTextAreaElement Interface
This interface is used for the textarea element. It is an extension to HTMLTextAreaElement.
- BMLTableCaptionElement Interface
This interface is used for the caption element. It is an extension to HTMLTableCaptionElement.
- BMLTableColElement Interface
This interface is used for the col and colgroup elements. It is an extension to HTMLTableColElement.
- BMLTableElement Interface
This interface is used for the table element. It is an extension to HTMLTableElement.
- BMLTableSectionElement Interface
This interface is used for the thead, tfoot, and tbody elements. It is an extension to HTMLTableSectionElement.
- BMLTableCellElement Interface
This interface is used for the th and td elements. It is an extension to HTMLTableCellElement.
- BMLTableRowElement Interface
This interface is used for the tr element. It is an extension to HTMLTableRowElement.
- BMLImageElement Interface
This interface is used for the img element. It is an extension to HTMLImageElement.

- BMLAreaElement Interface

This interface is used for the area element. It is an extension to HTMLAreaElement.

- BMLMapElement Interface

This interface is used for the map element. It is an extension to HTMLMapElement.

- BMLFrameSetElement Interface

This interface is used for the frameset element.
It is an extension to HTMLFrameSetElement.

- BMLFrameElement Interface

This interface is used for the frame element. It is an extension to HTMLFrameElement.

- BMLIFrameElement Interface

This interface is used for the iframe element. It is an extension to HTMLIFrameElement.

(2) Interfaces for Elements with Extended Attributes for Broadcasting Service

The following interfaces are the HTML DOM interface defined in Section 7.1.3 with extensions for attributes and CSS properties added in a BML element for broadcasting service.

- BMLDivElement Interface

This interface is used for the div element. It corresponds to the additional definition of the attributes the for div element in Section 5.3.1.2.

Interface definition:

```
interface BMLDivElement : HTMLDivElement {
    attribute DOMString          accessKey;
    attribute BMLCSS2Properties   style;
    attribute BMLCSS2Properties   normalStyle;
    attribute BMLCSS2Properties   focusStyle;
    attribute BMLCSS2Properties   activeStyle;
    void focus();
    void blur();
};
```

Attributes:

accessKey	Value of the accesskey attribute
style	Value of CSS property set to the style 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

- BMLParagraphElement Interface

This interface is used for the p element. It corresponds to the additional definition of attributes for the p element in Section 5.3.1.2.

Interface definition:

```
interface BMLParagraphElement : HTMLParagraphElement {
    attribute DOMString          accessKey;
    attribute BMLCSS2Properties   style;
    attribute BMLCSS2Properties   normalStyle;
    attribute BMLCSS2Properties   focusStyle;
    attribute BMLCSS2Properties   activeStyle;
    void      focus();
    void      blur();
};
```

Attributes:

accessKey	Value of the accesskey attribute
style	Value of CSS property set to the style 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

- BMLSpanElement Interface

This interface is used for the span element. It corresponds to the additional definition of the attributes for the span element in Section 5.3.1.2.

Interface definition:

```
interface BMLSpanElement : HTMLSpanElement {
    attribute DOMString          accessKey;
    attribute BMLCSS2Properties   style;
    attribute BMLCSS2Properties   normalStyle;
    attribute BMLCSS2Properties   focusStyle;
    attribute BMLCSS2Properties   activeStyle;
```

```
void    focus();
void    blur();
};
```

Attributes:

accessKey	Value of the accesskey attribute
style	Value of CSS property set to the style 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

- BMLAnchorElement Interface

This interface is used for the a element. It corresponds to the additional definition of attributes for the a element in Section 5.3.1.3.

Interface definition:

```
interface BMLAnchorElement : HTMLAnchorElement {
    attribute BMLCSS2Properties    style;
    attribute BMLCSS2Properties    normalStyle;
    attribute BMLCSS2Properties    focusStyle;
    attribute BMLCSS2Properties    activeStyle;
    attribute BMLCSS2Properties    effect;
};
```

Attributes:

style	Value of CSS property set to the style 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.
effect	Special effect for screen transition. Value of effect attribute of the a element (See section 5.3.20.3).

Method:

None

- BMLFormElement Interface

This interface is used for the form element. It is HTMLFormElement interface defined in Section 7.1.3 with an interface with the accept attribute defined in XHTML 1.0 (Undefined in HTML DOM).

Interface definition:

```
interface BMLFormElement : HTMLFormElement {
    attribute BMLCSS2Properties style;
    attribute BMLCSS2Properties normalStyle;
    attribute BMLCSS2Properties focusStyle;
    attribute BMLCSS2Properties activeStyle;
    attribute DOMString accept;
};
```

Attributes:

style	Value of CSS property set to the style 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.
accept	Specifies Content-Type list received from the server. Value of the accept attribute of the form element.

Methods:

None

- BMLObjectElement 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 in Section 5.3.20.5.

Interface definition:

```
interface BMLObjectElement : HTMLObjectElement
    attribute BMLCSS2Properties style;
    attribute BMLCSS2Properties normalStyle;
    attribute BMLCSS2Properties focusStyle;
    attribute BMLCSS2Properties activeStyle;
    attribute DOMString classId;
    attribute boolean remain;
    attribute long streamPosition;
    attribute DOMString streamStatus;
    attribute long streamLooping;
    attribute long streamSpeedNumerator;
    attribute long streamSpeedDenominator;
    attribute DOMString streamLevel;
    attribute DOMString mainAudioStreamI;
```

```

boolean    setSpeed (input long numerator, input long denominator);
boolean    movePosition(input long offset);
boolean    selectMainAudioStream(input DOMString audio_ref);
boolean    hasAssociatedIndex();
boolean    assignToLocalEvent(input long local_event_id);
boolean    assignToNodePlayMode(input DOMString node_ref);
attribute  DOMString          accessKey;
void       focus();
void       blur();
};

```

Attributes:

classId	Specifies the identifier that indicates the location of the object. Value of the classId attribute of the object element
style	Value of CSS property set to the style 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 for presentation in active state.
remain	If true, continues monomedia play while document transition. Value of the remain attribute (see Section 5.3.20.2) of the object element
streamPosition	Relative position of play to the head of the stream. Value of the streamposition attribute (see Section 5.3.16.2) 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 (see Section 5.3.20.2) of the object element
streamLooping	Number of repeated plays of stream. Value of the streamlooping attribute (see Section 5.3.20.2) of the object element
streamSpeedNumerator	Play speed (numerator). It represents the play speed in conjunction with the streamSpeedDenominator (denominator). Value of the streamspeednumerator attribute (see Section 5.3.20.2) of object element
streamSpeedDenominator	Play speed (denominator). It represents the play speed in conjunction with the streamSpeedNumerator (numerator). Value of the streamspeeddenominator attribute (see Section 5.3.20.2) the of object element
streamLevel	Loudness level of the audio stream. Value of the streamlevel attribute (see Section 5.3.20.2) of the object element
accessKey	Value of the accesskey attribute

Methods:

setSpeed

Specifies both the numerator and denominator of the playback speed of stream.

Parameter:

numerator	Playback speed (numerator)
denominator	Playback speed (denominator)

Return value:

true for success and false for fail.

movePosition

Specifies the relative current position for stream playback.

Parameter:

offset	Relative current position for stream playback
--------	---

Return value:

true for success and false for fail.

setMainAudioStream

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 (see Description 2).

Parameter:

audio_ref	URI character string indicating audio ES/channel described in the following format: /<component_tag>[;<channel_id>/]
-----------	---

Return value:

true for success and false for fail.

getMainAudioStream

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.

hasAssociatedIndex

Applicable to a stream in a storage device. The method verifies that a corresponding local event indexes¹ are recorded.

Parameter:

None

Return value:

If corresponding local event indexes exist, true. If there are no such indexes, false.

assignToLocalEvent

¹ When storing a stream, it is assumed that local event index is recorded in pairs with the stream. Therefore, there is no need for an API for relating each stream with a local event index or assignment of such a relation in the name space.

Applicable to a stream in a storage device and its local event index. The method specifies a local event in LIT included in the local event index and sets the start and end of play positions for play of the stream that represents the local event.

Parameters:

local_event_id Local event identifier

Return values:

true for success and false for fail.

assignToNodePlayEvent

Applicable to a stream in a storage device and its local event indexes. The method specifies a node in ERT that is contained in the local event indexes. Then it sets up so that all local events that reference the node are played in the order that is specified by the reference descriptor of the LIT. The start of play position is set to the beginning of the first local event. The end of play position is set to the end of the last local event.

Parameter:

node_ref

URI character strings indicating a node in ERT.
It is described in the following format.

/<information_provider_id>/<event_relation_id>/<node_id>

Return value:

true for success and false for fail.

focus Moves the focus to the item.

Parameter: None

Return value: None

blur Moves the focus away from the item.

Parameters: None

Return values: None

Applicability of the attributes and methods described above is according to Content-Type as shown in Table 7-2. Attributes and methods that are meaningless to the monomedia must be able to read and write attributes, and execute methods normally.

Table 7-2 Applicability of attribute and methods

Content-Type	video/****	audio/****	image/****	Others
attributes				
remain	O (Note 2)	O (Note 2)	O (Note 1)	O
streamPosition	O	O	O (Note 3)	-
streamStatus	O	O	O (Note 3)	-
streamLooping	O	O	O (Note 3)	-
streamSpeedNumerator	O	O	O (Note 3)	-
streamSpeedDenominator	O	O	O (Note 3)	-
streamStartPosition	O	O	O (Note 3)	-
streamEndPosition	O	O	O (Note 3)	-
streamLevel	-	O	-	-
methods				
setSpeed()	O	O	O (Note 3)	-

Content-Type	video/****	audio/****	image/****	Others
movePosition()	O	O	O (Note 3)	-
assignToLocalEvent()	O	O	-	-
assignToNodePlayMode()	O	O	-	-

Legend) O: Meaningful, -: Meaningless

Note 1: Two or more documents belonging to a common document group can share JPEG data that is stored in the content memory using lockModuleOnMemory().

Note 2: This data can be shared by documents in different ESs, as long as the ESs belong to a common content group.

Note 3: Only for X-arib-mng.

- BMLBodyElement Interface

This interface is used for the body element. It corresponds to the additional attribute definition of the body element in Section 5.3.20.4.

Interface definition:

```
interface BMLBodyElement : HTMLBodyElement {
    attribute BMLCSS2Properties      style;
    attribute BMLCSS2Properties      normalStyle;
    attribute BMLCSS2Properties      focusStyle;
    attribute BMLCSS2Properties      activeStyle;
    attribute boolean                invisible;
}
```

Attributes:

style	Contains the value of a CSS property that is set as the style attribute.
normalStyle	Contains the inherited value of a CSS property that is applied for presentation in normal state.
focusStyle	Contains the inherited value of a CSS property that is applied for presentation in focus state.
activeStyle	Contains the inherited value of a CSS property that is applied for presentation in active state.
invisible	When it is true, no element and no background of the BML document is displayed.

Methods:

None

(3) Interfaces for BML Elements Newly Defined for Broadcasting Service

The following interfaces are used for the BML elements additionally defined for broadcasting service.

- BMLBmlElement Interface

This interface is used for the bml element. The bml element corresponds to the html element in HTML.

Interface definition:

```
interface BMLBmlElement : HTMLHtmlElement {
    attribute BMLCSS2Properties      style;
```

```

        attribute BMLCSS2Properties          normalStyle;
        attribute BMLCSS2Properties          focusStyle;
        attribute BMLCSS2Properties          activeStyle;
    };

```

Attributes:

style	Contains the value of a CSS property that is set as the style attribute.
normalStyle	Contains the inherited value of a CSS property that is applied for presentation in normal state.
focusStyle	Contains the inherited value of a CSS property that is applied for presentation in focus state.
activeStyle	Contains the inherited value of a CSS property that is applied for presentation in active state.

Methods:

None

- BMLBeventElement Interface

This interface is used for the bevent element, which is an extended BML element for specifying events defined in Section 5.3.20.1.

Interface definition:

```

interface BMLBeventElement : HTMLElement {
};

```

Attributes:

None

Methods:

None

- BMLBeitemElement Interface

This interface is used for the beitem element, which is an extended BML element for specifying events defined in Section 5.3.20.1.

Interface definition:

```

interface BMLBeitemElement : 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 unsigned short              registerId;
    attribute unsigned short              serviceId;
    attribute unsigned short              eventId;
};

```

```

        attribute DOMString          timeMode;
        attribute DOMString          timeValue;
        attribute DOMString          objectId;
        attribute DOMString          segmentId;
        attribute boolean             subscribe;
    };

```

Attributes:

type	Type of events. Value of type attribute (see Section 5.3.20.1) of beitem element
esRef	Value of es_ref attribute (see Section 5.3.20.1) of beitem element
MessageId	Value of message_id attribute (see Section 5.3.20.1) of beitem element
MessageVersion	Value of message_version attribute (see Section 5.3.20.1) of beitem element
MessageGroupId	Value of message_group_id attribute (see Section 5.3.20.1) of beitem element
moduleRef	Value of module_ref attribute (see Section 5.3.20.1) of beitem element
languageTag	Value of language_tag attribute (see Section 5.3.20.1) of beitem element
registerId	Value of register_id attribute (see Section 5.3.20.1) of beitem element
serviceId	Value of service_id attribute (see Section 5.3.20.1) of beitem element
eventId	Value of event_id attribute (see Section 5.3.20.1) of beitem element
timeMode	Value of time_mode attribute (see Section 5.3.20.1) of beitem element
timeValue	Value of time_value attribute (see Section 5.3.20.1) of beitem element
objectId	Value of object_id attribute (see Section 5.3.20.1) of beitem element
subscribe	Value of subscribe attribute (see Section 5.3.20.1) of beitem element. Specifies whether events are valid or not.

Methods:

None

7.2 Scripting Language

This section defines the Scripting Language used in BML documents. The Scripting Language is based on ECMAScript that is defined in ECMA-262.

7.2.1 Base conventions

The **Syntax** and semantics, and built-in objects of the scripting language conform to ECMA-262. However, the following exceptions are allowed in real operation.

- 1) Using EUC-JP instead of JIS X 0221 for character encoding.
- 2) Setting the Number size smaller than 64 bits.
- 3) Not implementing Float.

7.2.2 Additional conventions

Broadcasting Extended APIs including date/time, table operations, external characters, EPG, operation control, event acquisition, and timer are added. Conventions for Broadcasting Extended Object Group and Browser Pseudo Objects are also added.

- Broadcasting Extended Object Group

This object group handles table data. It includes CSVTable object and BinaryTable object. Its operation is the same as the native objects of ECMAScript.

- Browser Pseudo Objects

These objects have been added to implement functions specific to broadcasting. Unlike the native objects, they do not inherit functions. They have global scopes.

7.2.3 Language binding

This section defines the bindings for accessing attributes and methods of DOM objects from an ECMAScript. The DOM and ECMAScript bindings conform to the DOM Level 1 Specifications, Appendix E “ECMAScript Language Binding.” In BML, the following types are bound.

Table 7-3 Binding of ECMA Script type and DOM-API type

ECMAScript Types	DOM-API Types
Number type	short int long unsigned long
String type	DOMString String
Boolean type	boolean
Object type	Other DOM objects

The root of the tree structure of an entire document is the Document object that has a BMLDocument interface or an HTMLDocument interface. It can be globally referenced from ECMAScript as an object that has the name “document.”

7.3 Security for Content

BML documents allow to use the Scripting Language to modify the configuration in a receiver or manipulate files on a receiver that has a storage device. To prevent an accidental or casual content from damaging the receiver and causing end users to lose their benefit, the two-tier security scheme described in this section is employed.

Class A	The contents that can invoke all functions including file management functions and system management functions.
Class B	The contents that cannot any file management function and system management function. They can invoke only the functions that are not likely to damage a receiver nor cause end users to lose their benefit.

Actual definition of Class A and Class B is responsible for an operational standard regulation for each media type, depending on the media characteristic and the broadcaster. Note that unless the harmlessness of any content is verified before it is broadcast, a security assurance method including an attached certificate should be required to identify a content as Class A.

7.4 Native Objects

The built-in objects of ECMAScript conform to the definitions of ECMA-262.

As stated before, EUC-JP as well as JIS X 0221 is allowed as character encoding in real operation. The character encoding of String objects used in a script must match the character encoding in that the script is written, which is the character encoding of the BML document.

The native objects are applicable to both any Class A content and Class B content.

7.5 Extended Object for Broadcasting

The extended objects for broadcasting defined in this section are applicable to both any Class A content and Class B content.

7.5.1 CSVTable object

This object receives two-dimensional table data and obtains a necessary sub-table from the data. The data consists of character strings in tabular format with line delimiting characters and column delimiting characters.

7.5.1.1 Data handled by CSVTable object

The table data handled by a CSVTable object is a file of character strings in a format with lines delimited by line feed characters and columns delimited by delimiter characters. The character encoding of characters in the table is EUC-JP, JIS X0221, or Shift-JIS.

7.5.1.2 Constructor of CSVTable object

- new CSVTable(): Generates a CSVTable object.

Syntax

```
CSVTable new CSVTable(  
    input String table_ref,  
    input Number delimiter  
)
```

Arguments

table_ref	Specifies a table file.
delimiter	Delimiter character

Return values

Generated CSVTable object
(If no object is generated, null.)

Description

This object generates a CSVTable object. The file specified by table_ref can be handled as a table until close() is performed. The [[Prototype]] property of the generated object is a built-in CSVTable prototype object. The [[Class]] property of the generated object is "CSVTable."

The description of table_ref conforms to the namespace conventions defined in Chapter 9.

The delimiter argument is used to specify delimiters. The following table shows the semantics of the delimiter argument.

Table 7-4 Values Applicable to delimiter

<delimiter>	Delimiter	<delimiter>	Delimiter
1	0x09 (Control code APF)	19	0x2F (Alphanumeric "/")
2	0x1E (Control code RS)	20	0x3A (Alphanumeric ":")
3	0x1F (Control code US)	21	0x3B (Alphanumeric ";")
4	0x20 (Control code SP)	22	0x3C (Alphanumeric "<")
5	0x21 (Alphanumeric "!")	23	0x3D (Alphanumeric "=")
6	0x22 (Alphanumeric "\"")	24	0x3E (Alphanumeric ">")
7	0x23 (Alphanumeric "#")	25	0x3F (Alphanumeric "?")
8	0x24 (Alphanumeric "\$")	26	0x40 (Alphanumeric "@")
9	0x25 (Alphanumeric "%")	27	0x5B (Alphanumeric "[")
10	0x26 (Alphanumeric "&")	28	0x5C (Alphanumeric "\")
11	0x27 (Alphanumeric "\"")	29	0x5D (Alphanumeric "]")
12	0x28 (Alphanumeric "(")	30	0x5E (Alphanumeric "^")
13	0x29 (Alphanumeric ")")	31	0x5F (Alphanumeric "_")
14	0x2A (Alphanumeric "*")	32	0x60 (Alphanumeric "`")
15	0x2B (Alphanumeric "+")	33	0x7B (Alphanumeric "{")
16	0x2C (Alphanumeric ",")	34	0x7C (Alphanumeric " ")
17	0x2D (Alphanumeric "-")	35	0x7D (Alphanumeric "}")
18	0x2E (Alphanumeric ".")	36	0x7E (Alphanumeric "~")

7.5.1.3 Property of CSVTable constructor

- CSVTable.prototype: Prototype

The initial value of CSVTable.prototype is a built-in CSVTable prototype object (see Section 7.5.1.4).

This property has the DontEnum, DontDelete, and ReadOnly attributes.

7.5.1.4 Property of CSVTable prototype object

The CSVTable prototype object itself is a CSVTable object. Its value is NaN.

The [[Prototype]] property of the CSVTable prototype object is an Object prototype object.

- CSVTable.prototype.constructor: Constructor

The initial value is a built-in CSVTable constructor (see Section 7.5.1.2).

- CSVTable.prototype.close(): Declares the end of the handling of CSVTable objects.

Syntax

Number CSVTable.prototype.close()

Argument

None

Return values

1: Success

NaN: Failure

Description

This object releases the memory and other areas for the table.

- CSVTable.prototype.toString(): Outputs an element of a table as a character string.

Syntax

String CSVTable.prototype.toString(input Number row, input Number column)

Arguments

row	Row
column	Column

Return values

Character string in a table element: Success

null: Specified element does not exist.

Description

This object returns an element of a table specified by a CSVTable object as a character string. The row and column arguments are numbers that are 0 (zero) or greater than zero indicating the location of the element.

- CSVTable.prototype.toNumber(): Outputs an element of a table as a numeric value.

Syntax

Number CSVTable.prototype.toNumber(
 input Number row,
 input Number column
)

Arguments

row	Row
column	Column

Return values

Numeric value of a table element: Success

NaN: Specified element does not exist.

Description

This object returns an element of a table specified by a CSVTable object as a numeric value. The row and column arguments are numbers that are 0 (zero) or greater than zero indicating the location of the element. The conversion from the character string in a table element into a number is equivalent to ToNumber().

- CSVTable.prototype.toArray(): Outputs a sub-table.

Syntax

Array CSVTable.prototype.toArray(input Boolean byRow,
 input Number startRow,
 input Number numRows,
 input Number startColumn,

input Number numColumn

)

Arguments

byRow	Direction of extraction (true: rows, false: columns)
startRow	Row (or column) to be extracted first
numRow	Number of rows to be extracted
startColumn	Column to be extracted first
numColumn	Number of columns to be extracted

Return values

Array objects corresponding to the extracted subtable:	Success
null:	Failure

Description

This object returns a sub-table of the table specified by a CSVTable object. The sub-table consists of adjacent rows or columns. Each element of the returned Array object is an Array object that represents each row or column of the sub-table. Each element of the Array object that represents each row or column is a String object.

Even if some or all of the specified elements do not exist in the table, this method returns an Array object (its elements are numRows number of Array objects). However, Array objects that have no corresponding rows or columns in the table and String objects that have no corresponding elements have a value of null.

- CSVTable.prototype.search(): Outputs rows or columns that satisfy one or more given conditions.

Syntax

Number CSVTable.prototype.search(

input Boolean byRow,
input Number startIndex,
[input Number searchedIndex,
input Object compared,
input Number operator]+,
input Boolean logic,
input Number limitCount,
output Array resultArray

)

Arguments

byRow	Direction of extraction (true: rows, false: columns)
startIndex	Row or column with which searching begins
searchedIndex	Row or column to be searched
compared	Compared to (String or Number type)
operator	Condition for comparison
logic	Relationship among more than one comparison condition (true: OR, false: AND)

limitCount	Maximum number of rows or columns to be extracted by searching
resultArray	Array to store the output rows and columns

Return values

Row or column with which the searching ended: Success

-1: All records were searched before reaching the limitCount number.

NaN: Failure

Description

This object returns rows or columns in a table specified by a CSVTable object that satisfy the search conditions. The result is returned in resultArray. The search conditions are specified by comparison conditions indicated by one or any combination of searchedIndex, compared, and operator, and by logic that shows the relationship among them.

Each element of resultArray is an Array object that represents each row or column that satisfies the search conditions. Each element of the Array object is a String object.

The search starts with startIndex and ends when up to limitCount number of rows or columns have been extracted. And the index of the row (or column) that was extracted last is returned. If all records have been searched before reaching limitCount, '-1' is returned.

The searchedIndex, compared, and operator arguments are variable argument lists. A set of these three arguments must be specified together. The operator argument is defined as follows. The type of compared is obtained by checking the value of the operator argument.

The values applicable to <operator> and their semantics when the type of <compared> is Number:

0:	Equal (==)	<compared>
1:	Not equal (≠)	<compared>
2:	Less than (<)	<compared>
3:	Less than or equal to (<=)	<compared>
4:	Greater than (>)	<compared>
5:	Greater than or equal to (>=)	<compared>
6:	&	<compared> (Bitwise AND logical operation)
7:		<compared> (Bitwise OR logical operation)
8:	^	<compared> (Bitwise Exclusive OR logical operation)
9:	~&	<compared> (Bitwise one's complement AND logical operation)
10:	~	<compared> (Bitwise one's complement OR logical operation)
11:	~^	<compared> (Bitwise one's complement Exclusive OR logical operation)

The values applicable to <operator> and their semantics when the type of <compared> is String:

32:	Matches <compared>
-----	--------------------

- 33: Includes <compared>
- 34: Starts with <compared>
- 35: Ends with <compared>
- 36: Does not match <compared>
- 37: Does not include <compared>

7.5.1.5 Property of CSVTable instance

The CSVTable instance inherits the properties of a CSVTable prototype object, and retains the [[Value]], nrow, and ncolumn properties.

- nrow: Number of rows

The nrow property indicates the number of rows in the table specified by the CSVTable object.

- ncolumn: Number of columns

The ncolumn property indicates the number of columns in the table specified by the CSVTable object.

7.5.2 BinaryTable object

This object receives two-dimensional table data and obtains a required sub-table from the data. The data is in binary representation.

7.5.2.1 Data handled by BinaryTable object

The table data handled by a BinaryTable object is a file that consists of one or a series of fixed or variable length records (rows). Each record consists of one or a series of fields (elements).

A variable record has an area that indicates the length of the record at the beginning of the record and at least one variable length field. A variable length field has an area of one or more bytes that indicates the length of the field at the beginning of the field.

The encoding scheme of characters in the table is EUC-JP, JIS X0221, or Shift-JIS.

7.5.2.2 Constructor of BinaryTable object

- Constructor: Generates a BinaryTable object.

Syntax

```
BinaryTable new BinaryTable(  
    input String table_ref,  
    input String structure  
)
```

Arguments

table_ref	Specifies a table file.
structure	Specifies the table format.

Return values

BinaryTable object generated:	Success
-------------------------------	---------

null: Generation failed

Description

This constructor generates a BinaryTable object. It handles a file specified by table_ref in a format specified by structure.

The [[Prototype]] property of the generated object is a built-in BinaryTable prototype object, which is the initial value of BinaryTable.prototype.

The [[Class]] property of the generated object is "BinaryTable."

The description of table_ref conforms to the namespace conventions defined in Chapter 9. This object is effective until close() is performed.

Format definitions of structure

structure ::= lengthByte ',' field[',' field]*

lengthByte ::= decint | "0"

field ::= type ":" size

type ::= "B" | "U" | "I" | "S" | "Z" | "P"

size ::= length unit

length ::= decint

decint ::= digit [digit0]*

digit0 ::= digit | "0"

digit ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

unit ::= "B" | "b" | "V"

lengthByte A value representing record length in bytes. 0 (zero) indicates that all fields are fixed length.

size If unit is "B" or "b," field length.
If unit is "V" or "v," number of bytes that contain field length. The table contains a field length followed by field data. For variable length fields, type is "S" or "Z." If the table contains the record length and field length, it must be unsigned integer. It is encoded in the network byte order (bigendian). The length of data following the length field is specified in bytes.

length A decimal number larger than 1.

unit Unit used for expressing the field length in length

"B" ... Byte	In bytes
"b" ... bit	In bits
"V" ... variable	Variable field

type Type of field

"B" ... Boolean

Indicates Boolean type data. It has 1 bit length. 0 indicates false and 1 indicates true. The unit must be set to bit ("b"). Handled as a Boolean object in procedural description language.

"U" ... Unsigned Integer

Indicates an unsigned integer. It has up to 4 bytes (32 bits) length. It is encoded in network byte order (bigendian). If it has 4 bytes (32 bits) length, the most significant bit must be 0 (The maximum allowable number is 2147483647(0x7FFFFFFF)). If it has 8 bits length or longer, the second and

later bytes must be in byte aligned. The unit must be set to Byte (“B”) or bit (“b”). Handled as a Number object in procedural description language.

If it has 7 bits length or shorter, it is not necessary to be byte-aligned.

“I” ... Integer

Indicates a signed integer. It has up to 4 bytes (32 bits) length. It is encoded in network byte order (bigendian). Negative values are expressed in two’s complements. It must be byte aligned. The unit must be set to Byte (“B”). It must be 1, 2, or 4 bytes length. Handled as a Number object in procedural description language.

“S” ... String

Indicates String type data. It must be byte aligned. The unit must be set to Byte (“B”) or variable (“V”). Handled as a String object in procedural description language.

“Z” ... ZipCode

Indicates a postal code (zip code) encoded data(It is specified in Annex B. The length of structure is length specified in Annex B.1 and itself). It must be byte aligned. The unit must be set to variable (“V”).

Handled as a Boolean object in procedural description language.

“P” ... Pad

Indicates a gap between data. It is not handled as a field. It is used to byte-align fields. The unit is set to Byte (“B”) or bit (“b”).

Sample format definition

"1,B:1b,B:1b,U:3b,U:3b,S:1V,Z:1V"

One byte record length is placed at the beginning of each record.

Following the record length, the following fields are placed in the record in the order listed.

Boolean type (1 bit)

Boolean type (1 bit)

Unsigned integer (3 bits)

Unsigned integer (3 bits)

Character string (variable length) Length data (1 byte)

Zip code (variable length) Length data (1 byte)

7.5.2.3 Properties of BinaryTable constructor

- BinaryTable.prototype: Prototype

The initial value of BinaryTable.prototype is a built-in BinaryTable prototype object (see Section 7.5.2.4).

This property has the DontEnum, DontDelete, and ReadOnly attributes.

7.5.2.4 Properties of BinaryTable prototype object

The BinaryTable prototype object itself is a BinaryTable object. Its value is NaN.

The [[Prototype]] property of a BinaryTable prototype object is an Object prototype object.

- BinaryTable.prototype.constructor: Constructor
The initial value is a built-in BinaryTable constructor (see Section 7.5.2.2).
- BinaryTable.prototype.close(): Declares the end of handling BinaryTable objects.

Syntax

Number BinaryTable.prototype.close()

Argument

None

Return values

1: Success

NaN: Failure

Description

This method releases memory and other areas for the table.

- BinaryTable.prototype.toString(): Outputs one field of a table as a character string.

Syntax

```
String BinaryTable.prototype.toString(  
    input Number row,  
    input Number column  
)
```

Arguments

row	Row
column	Column

Return values

Character string in a table field: Success

null: Specified field does not exist.

Description

This method returns one field of a table specified by a BinaryTable object as a character string. The row and column arguments are numbers that are 0 (zero) or greater than zero that indicating the location of the field. If the field content is not a character string, the result of conversion using toString() is returned. However, if the field contains a ZipCode, null is returned.

- BinaryTable.prototype.toNumber(): Outputs a field of a table as a numeric value.

Syntax

```
Number BinaryTable.prototype.toNumber(  
    input Number row,  
    input Number column  
)
```

Arguments

row	Row
column	Column

Return values

Numeric value a table field:	Success
NaN:	Specified field does not exist.

Description

This method returns one field of a table specified by a BinaryTable object as a numeric value. The row and column are numbers that are 0 (zero) or greater than zero indicating the location of the field. If the specified field does not exist in the table, NaN is returned.

If the field content is not an integer, the result of conversion using toNumber() is returned. However, if the field contains a ZipCode, NaN is returned.

- BinaryTable.prototype.toArray(): Outputs records in a table in Array.

Syntax

```
Array BinaryTable.prototype.toArray(input Number startRow,
input Number numRows)
```

Arguments

startRow	Record with which extraction begins
numRows	Number of records to be extracted

Return values

Array storing a series of records:	Success
null:	Failure

Description

This method extracts a adjacent records in a table specified by a BinaryTable object as Array objects, then returns an Array object whose elements are those Array objects.

Each element of the Array object that represents each record is an object with the type same as the field. However, for ZipCode fields, null is returned.

The numRows number of records starting with the startRow-th record (that must be 0 or greater) are extracted.

If some or all of the specified records do not exist in the table, this method still returns an Array object (its elements are numRows number of Array objects). However, Array objects that have no corresponding records in the table have a value of null.

- BinaryTable.prototype.search(): Outputs records in a table that satisfy one or more conditions.

Syntax

```
Number BinaryTable.prototype.search(
input Number startRow,
[input Number searchedColumn,
input Object compared,
input Number operator] +,
input Boolean logic,
input Number limitCount,
output Array resultArray
)
```

Arguments

startRow	Record with which searching starts.
searchedColumn	Column to be searched
compared	Compared to (String, Number, or Boolean type)
operator	Condition for comparison
logic	More than one comparison condition (true: OR, false: AND)
limitCount	Maximum number of records to be extracted by searching
resultArray	Array to store the records

Return values

Record with which the search ended: Success

-1: All records are searched before reaching the limitCount number

NaN: Failure

Description

This method returns records in a table specified by a BinaryTable object that satisfy the search conditions. The search conditions are specified by comparison conditions indicated by one or more sets of searchedColumn, compared, and operator, and by logic that shows the relationship between them. The broadcaster specifies the maximum number of conditions for comparison.

The result is returned in resultArray. Each element of resultArray is an Array object that represents each record satisfying the search conditions. Each element of the Array object is an object with the type of the field. However, the following is applicable to the element of Array object corresponding to the ZipCode type field: When comparison conditions for this field has been designated, the result of judging based on the designated comparison condition is output as a Boolean object. When no comparison condition has been designated, null is set.

The search starts at startRow and ends when up to the “limitCount” number of records have been extracted. The index of the last record extracted is returned. If all records have been searched before reaching limitCount, ‘-1’ is returned.

The searchedColumn argument, the compared argument, and the operator argument are variable argument lists. These three arguments must be specified together. The operator argument is defined as follows. Type of compared is obtained by checking the value of operator. However, the compared argument that is compared with ZipCode is a Number object, which is treated as a decimal 7-digit integer. When compared is a String object, searching works by comparing with a character string. When compared is a Number object, searching works by comparing with a signed integer.

The values applicable to operator and their semantics when the field type indicated by searchedColumn is Unsigned Integer or Integer.:

0:	(=)	<compared>
1:	(≠)	<compared>
2:	(<)	<compared>
3:	(<=)	<compared>
4:	(>)	<compared>
5:	(>=)	<compared>
6:	&	<compared> (Bitwise AND logical operation)
7:		<compared> (Bitwise OR logical operation)

- 8: ^ <compared> (Bitwise Exclusive OR logical operation)
- 9: ~& <compared> (Bitwise one's complement AND logical operation)
- 10: ~| <compared> (Bitwise one's complement OR logical operation)
- 11: ~^ <compared> (Bitwise one's complement Exclusive OR logical operation)

The values applicable to operator and their semantics when the field type of searchedColumn is String:

- 32: Matches <compared>
- 33: Includes <compared>
- 34: Starts with <compared>
- 35: Ends with <compared>
- 36: Does not match <compared>
- 37: Does not include <compared>

The values applicable to operator and their semantics when the field type of searchedColumn is Boolean:

- 64: Equal to <compared>
- 65: Not equal to <compared>

The values applicable to operator and their semantics when the field type of searchedColumn is ZipCode:

- 96: Includes a zip code specified in <compared>
- 97: Does not include a zip code specified in <compared>

Note: When FieldType is Unsigned Integer or Integer, and the operator is "&", "|", "^", "~&", "~|", or "~^", the decision is made according to whether the result is 0 or not. If it is 0, the return value is false, otherwise true.

7.5.2.5 Properties of BinaryTable Instance

The BinaryTable instance inherits the properties of BinaryTable prototype object, and retains the [[Value]], nrow, and ncolumn properties.

- nrow: Number of rows

The nrow property indicates the number of rows in the table specified by the BinaryTable object.

- ncolumn: Number of columns

The ncolumn property indicates the number of columns in the table specified by the BinaryTable object.

7.5.3 XML document Object

An XML document object works in a BML object. An XML document Object is used to import an XML document to be accessed by a DOM with ECMAScript. An XML document Object is also used to export a DOM tree as an XML document to an external file or external device.

For the purpose of this section, the term "External XML document" means an XML document that is used based on the definitions in this section to explicitly distinguish from an XML document used as defined in Chapter 6.

7.5.3.1 Data handled by XML document Object

An External XML document which is handled by an XML document manipulates a well-formed document complying with XML 1.0; referencing to DTD and obtaining DTD is not required. The character encoding complies with Section 4.1 in this standard. Note that an External XML document must use the same character encoding as that used by a BML document referencing the External XML document.

7.5.3.2 Constructor of XML document Object

- constructor: Generates an XML document object.

Syntax

XMLDoc new XMLDoc()

Argument

None

Return values

Generated XMLDoc object:	success
null :	no object is generated

Description

This constructor generates an XMLDoc object. The [[Prototype]] property of the generated object is a built-in XMLDoc prototype object and contains the initial value of XMLDoc.prototype. The [[Class]] property of the generated object is "XMLDoc." This object is effective until close() is performed.

7.5.3.3 Property of XMLDoc Constructor

- XMLDoc.prototype: Prototype

The initial value of XMLDoc.prototype is a built-in XMLDoc prototype object. This property has the DontEnum, DontDelete, and ReadOnly attributes.

7.5.3.4 Property of XMLDoc Prototype Object

An XMLDoc prototype object itself is an XMLDoc object. The value of an XMLDoc prototype object is NaN. The [[Prototype]] property of an XMLDoc prototype object is an Object prototype object.

- XMLDoc.prototype.constructor: Constructor

The initial value is a built-in XMLDoc constructor.

- XMLDoc.prototype.close(): Declares the end of the handling of XMLDoc objects.

Syntax

Number XMLDoc.prototype.close()

Argument

None

Return values

1:	Success
----	---------

NaN: Failure

Description

This method destroys the DOM tree contained in the object to release the memory and other areas.

- XMLDoc.prototype.create(): Generates an empty DOM tree.

Syntax

Number XMLDoc.prototype.create()

Argument

None

Return values

1: Success

NaN: Failure

Description

This method creates an empty DOM tree. When the object contains a DOM tree, the existing DOM tree is deleted.

- XMLDoc.prototype.read(): Import an External XML document to create a DOM tree.

Syntax

```
Number XMLDoc.prototype.read(
    input String XMLDoc_ref
)
```

Argument

XMLDoc_ref: URI of the External XML document

Return values

1: Success

NaN: Failure

Description

This method imports the External XML document located in URI specified in XMLDoc_ref to create a DOM tree. When the object contains a DOM tree, the existing DOM tree is deleted and the External XML document specified in XMLDoc_ref is imported. XMLDoc_ref is described based on the namespace definitions, as specified in Chapter 9 in this standard.

- XMLDoc.prototype.getDocument(): Returns a DOM tree of an object as a Document object.

Syntax

Document XMLDoc.prototype.getDocument()

Argument

None

Return values

Document object: Success

null: Failure

Description

This method returns the DOM tree contained in the object as a Document object. When the object contains no DOM tree, null is returned.

- XMLDoc.prototype.write (): Creates an External XML document based on a DOM tree to export it to URI specified in XMLDoc_ref

Syntax

```
Number XMLDoc.prototype.write(  
    input String XMLDoc_ref,  
    input String encoding  
)
```

Arguments

XMLDoc_ref: URI of an XML document
Encoding: Character encoding used by the XML document to be created

Return values

1: Success
NaN: Failure

Description

This method exports the DOM tree contained in the object to URI specified in XMLDoc_ref. The DOM tree is written as an External XML document. XMLDoc_ref is described based on the namespace definitions, as specified in Chapter 9 in this standard. The encoding argument contains the character encoding used by the XML document to be created.

7.5.3.5 Property of XMLDoc Instance

The XMLDoc instance inherits the properties of an XMLDoc prototype object, and retains the [[Value]] property.

7.6 Extended Functions for Broadcasting (Browser Pseudo Object)

This section defines extended functions that do not have to retain data structures as objects. When these functions used to bind ECMAScript, they are treated as Browser Pseudo Objects. A Browser Pseudo Object is a global object that provides a broadcasting extended function as a method. It is accessed with a browser.method name or a browser.Ureg.

Functions starting with 'X' are used for extended functions proprietary defined by individual media type, broadcaster, terminal manufacturer, or other consortium. They are not used in the following definitions. To prevent conflict of function names with these functions, broadcasters must establish additional conventions on function names.

7.6.1 EPG functions

- 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 on namespace defined in Section 9.2.6.

- 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 on namespace defined in Section 9.2.6.

- epgTune(): Quits displaying the presented BML 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 service_ref conforms to the namespace defined in Section 9.2.5.

- The scripts after the epgTune() are not executed.
 - If epgTune() is executed in a global code, the “load” event and “unload” are not occurred.
 - If epgTrue() fails, it is not ensured that the following scripts are executed.
- epgTuneToComponent (): Quits displaying the presented BML document and selects the specified component.

Syntax

Number epgTuneToComponent(input String component_ref)

Argument

component_ref: Specifies the component to be selected

Return values

1: Success
NaN: Failure

Description

The description of component_ref conforms to the namespace conventions defined in Section 9.2.1.1. This object selects the service transmitting the data component specified in component_ref and launches the BML document to be transmitted via the specified data component. The BML document to be launched upon specifying the component is selected based on the criteria defined in Section 9.2.2. The other behaviours of this function are defined as below:

- The scripts following epgTuneToComponent() are not executed.
 - If epgTuneToComponent() is executed in the global code, the “load” event and the “unload” event are not occurred.
 - If epgTuneToComponent() fails, it is not ensured that the following scripts are executed.
 - As soon as the event handler invoking this API quits, a browser causes the onunload event to occur. Then, the event handler invoked by the onunload event is quitted before the content specified in component_ref is presented.
 - If this object fails to present the component via the selected service, it launches and presents a BML document that is selected from the default components for the selected service based on the criteria defined in Section 9.2.2. to present the BML document. If this object fails to present this default component, the following behaviours depends on an implementation.
- epgTuneToDocument (): Quits displaying the presented BML document and presents the specified BML document.

Syntax

Number epgTuneToDocument(input String documentName)

Argument

DocumentName: String specifying the BML document to be presented

Return values

1: Success

NaN: Failure

Description

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

- The scripts following epgTuneToDocument() are not executed.
 - If epgTuneToDocument() is executed in the global code, the “load” event and “unload” are not occurred
 - 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 namespace conventions defined in Section 9.2.6.

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 namespace conventions defined in Section 9.2.6.

If startTime is omitted, this function acts on the event specified by event_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 namespace conventions defined in Section 9.2.6.

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 namespace conventions defined in Section 9.2.6.

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 namespace conventions defined in Section 9.2.6.

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 a event specified by event_ref and returns the result of cancellation.

The description of event_ref conforms to the namespace conventions defined in Section 9.2.6.

7.6.2 Event group index functions

- grpIsReserved (): Verifies whether all events that reference a specified node are reserved for watching.

Syntax

Number grpIsReserved(input String node_ref)

Argument

node_ref Specifies a node.

Return values

- 1: All programs that reference a specified node are reserved for watching.
- 0: Some programs that reference a specified node are not reserved for watching.
- NaN: Failure

Description

This object verifies whether all events that reference a node specified by node_ref are reserved for watching and returns the result of verification. The description of node_ref conforms to the namespace conventions defined in Chapter 9.

- grpReserve (): Reserves for watching all events that reference a specified node.

Syntax

Number grpReserve(input String node_ref)

Argument

node_ref Specifies a node.

Return values

- 1: Success
- NaN: Failure

Description

This object reserves all events for watching which have referred to the node designated at node_ref. Success or failure is returned at the value. The description of node_ref conforms to the namespace conventions defined in Chapter 9.

- grpCancelReservation(): Cancels the reservation of all events for watching that reference a specified node.

Syntax

Number grpCancelReservation(input String node_ref)

Argument

node_ref Specifies a node.

Return values

- 1: Success

NaN: Failure

Description

Cancels the reservation of all events for watching that reference a node specified by `node_ref` and returns the result of cancellation. The description of `node_ref` conforms to the namespace conventions defined in Chapter 9.

- `grpRecIsReserved ()`: Verifies whether all events that reference a specified node are reserved for recording.

Syntax

Number `grpRecIsReserved(input String node_ref)`

Argument

`node_ref` Specifies a node.

Return values

1: All programs that reference a specified node are reserved for recording.
0: Some programs that reference a specified node are not reserved for recording.
NaN: Failure

Description

This object verifies whether or not all events that reference a node specified by `node_ref` are reserved for recording and returns the result of verification. The description of `node_ref` conforms to the namespace conventions defined in Chapter 9.

- `grpRecReserve()`: Reserves for recording all events that reference a specified node.

Syntax

Number `grpRecReserve(input String node_ref)`

Argument

`node_ref` Specifies a node.

Return values

1: Success
NaN: Failure

Description

Reserves all events for recording which have referred to the node designated at `node_ref`. Success or failure is returned at the value.

The description of `node_ref` conforms to the namespace conventions defined in Chapter 9.

- `grpRecCancelReservation()`: Cancels the reservation of all events for recording that reference a specified node.

Syntax

Number `grpRecCancelReservation(input String node_ref)`

Argument

`node_ref` Specifies a node.

Return values

1: Success
NaN: Failure

Description

This object cancels the recording reservation of all events which have referred to the node designated by node_ref. Success or failure is returned by the value.

The description of node_ref conforms to the namespace conventions defined in Chapter 9.

- grpGetNodeEventList(): Outputs a list of all events that reference a specified node.

Syntax

Array grpGetNodeEventList(input String node_ref)

Argument

node_ref Specifies a node.

Return values

Array object storing the result: Success

null: Failure

Description

This function obtains the list of the events referring to the node specified in argument node_ref as an array. When there are no applicable events, an array of length 0 is returned. When an array which has one or more elements is returned, each element is a string to represent an event of which description conforms to the namespace conventions defined in Chapter 9.

- grpGetERTNodeName (): Obtains the node name of a specified node in ERT.

Syntax

String grpGetERTNodeName(input String node_ref)

Argument

node_ref Specifies a node.

Return values

Node name: Success

null: Failure

Description

The description of node_ref conforms to the namespace conventions defined in Chapter 9. The content of “node_name_char” in the short form node information descriptor of a node specified by the node_ref argument is returned with the character codes converted. For the short form node information descriptor, see ARIB STD-B10 Version 1.2, Volume 3, 5.2.4.

- grpGetERTNodeDescription(): Obtains the node description of a specified node in ERT.

Syntax

String grpGetERTNodeDescription(input String node_ref)

Argument

node_ref Specifies a node.

Return values

Node description: Success

null: Failure

Description

The description of `node_ref` conforms to the namespace conventions defined in Chapter 9. The content of “`text_char`” in the short form node information descriptor of a node specified by the `node_ref` argument is returned with the character codes converted. For the short form node information descriptor, see ARIB STD-B10 Version 1.2, Volume 3, 5.2.4.

- `epgXTune()`: Quits displaying the presented BML document and invokes a receiver EPG function using an event index in a specified entry (ERT node). (Pass the control to the receiver EPG function using the event index that the receiver has.)

Syntax

Number `epgXTune(input String node_ref)`

Argument

<code>node_ref</code>	Specifies a node.
-----------------------	-------------------

Return values

1:	Success
NaN:	Failure

Description

The description of `node_ref` conforms to the namespace conventions defined in Chapter 9

7.6.3 Series reservation functions

This section specifies the reservation functions applicable to the series specified by the series descriptor².

- `seriesIsReserved()`: Verifies whether or not the specified series are reserved for viewing.

Syntax

Number `seriesIsReserved(input String series_ref, input Date expire_date)`

Arguments

<code>series_ref</code>	Specifies the series
<code>expire_date</code>	Term of validity for the series

Return values

1:	Reserved for viewing
0:	Not reserved
NaN:	Failure

Description

It verifies whether or not the series specified by `series_ref` is reserved for viewing and results the return value. However, if the absolute time at which this function is executed is after the time designated by `expire_date`, it decides that the specification of `series_ref` is invalid and returns NaN. Further, The description of `series_ref` conforms to the namespace conventions defined in Chapter 9.

- `seriesReserve()`: Reserves the specified series for viewing.

Syntax

² The series descriptor will be specified in ARIB STD-B10 Version 1.3.

Number seriesReserve(input String series_ref, input Date expire_date)

Arguments

series_ref	Specifies the series
expire_date	Term of validity for the series

Return values

1:	Success
NaN:	Failure

Description

It reserves the series specified by series_ref for viewing and returns the results with the return value. However, if the absolute time at which this function is executed is after the time designated by expire_date, it decides that the specification of series_ref is invalid and returns NaN. Further, The description of series_ref conforms to the namespace conventions defined in Chapter 9.

- seriesCancelReservation(): Cancels the reservation of the specified series for viewing.

Syntax

Number seriesCancelReservation(input String series_ref, input Date expire_date)

Arguments

series_ref	Specifies the series
expire_date	Term of validity for the series

Return values

1:	Success
NaN:	Failure

Description

It cancels the reservation of the series specified by series_ref for viewing and returns the results with the return value. However, if the absolute time at which this function is executed is after the time designated by expire_date, it decides that the specification of series_ref is invalid and returns NaN. Further, The description of series_ref conforms to the namespace conventions defined in Chapter 9.

- seriesRecIsReserved(): Verifies whether or not the specified series are reserved for recording.

Syntax

Number seriesRecIsReserved(input String series_ref, input Date expire_date)

Arguments

series_ref	Specifies the series
expire_date	Term of validity for the series

Return values

1:	Reserved for recording
0:	Not reserved
NaN:	Failure

Description

It verifies whether or not the series specified by `series_ref` is reserved for recording and results the return value. However, if the absolute time at which this function is executed is after the time designated by `expire_date`, it decides that the specification of `series_ref` is invalid and returns NaN. Further, The description of `series_ref` conforms to the namespace conventions defined in Chapter 9.

- `seriesRecReserve()`: Reserves the specified series for recording.

Syntax

Number `seriesRecReserve(input String series_ref, input Date expire_date)`

Arguments

<code>series_ref</code>	Specifies the series
<code>expire_date</code>	Term of validity for the series

Return values

1:	Success
NaN:	Failure

Description

It reserves the series specified by `series_ref` for recording and returns the results with the return value. However, if the absolute time at which this function is executed is after the time designated by `expire_date`, it decides that the specification of `series_ref` is invalid and returns NaN. Further, The description of `series_ref` conforms to the namespace conventions defined in Chapter 9.

- `seriesCancelRecReservation()`: Cancels the reservation of the specified series for recording.

Syntax

Number `seriesCancelRecReservation(input String series_ref, input Date expire_date)`

Arguments

<code>series_ref</code>	Specifies the series
<code>expire_date</code>	Term of validity for the series

Return values

1:	Success
NaN:	Failure

Description

It cancels the reservation of the series specified by `series_ref` for recording and returns the results with the return value. However, if the absolute time at which this function is executed is after the time designated by `expire_date`, it decides that the specification of `series_ref` is invalid and returns NaN. Further, The description of `series_ref` conforms to the namespace conventions defined in Chapter 9.

7.6.4 Subtitle presentation control functions

The following functions allow a BML document to control the display state of subtitles and to select a language. The display state of subtitles and language selection are assumed to be performed by the subscriber using features of a receiver, independent of the multimedia service using BML/B-XML.

The following group of functions can be used with CCStatusChanged of the beitem element to enable describing scripts in a BML document for controlling subtitle switching coordinated with the switching by the receiver.

- setCCStreamReference(): Selects a component stream of the subtitle.

Syntax

Number setCCStreamReference(input String stream_ref)

Argument

stream_ref URI to identify a component stream.

Return values

1: Success

NaN: Failure

Description

This function selects a component stream of the subtitle to be controlled and operated.

- getCCStreamReference(): Obtains the URI of the selected component stream of the subtitle.

Syntax

String getCCStreamReference()

Arguments

None

Return values

URI to identify a component stream: Success

null: Failure

Description

This function obtains the URI that identifies the component stream that transmits the subtitle currently displayed.

- setCCDisplayStatus(): Switches the display state of the specified language.

Syntax

Number setCCDisplayStatus(
 input Number language,
 input Boolean status
)

Arguments

language	Language selection
1:	1st language
2:	2nd language
3:	3rd language
4:	4th language
5:	5th language
6:	6th language

	7:	7th language
	8:	8th language
status	Display control	
	True:	Presents
	False:	Dose 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 do 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: 1st language
	2: 2nd language
	3: 3rd language
	4: 4th language
	5: 5th language
	6: 6th language
	7: 7th language
	8: 8th 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: 1st language
	2: 2nd language
	3: 3rd language
	4: 4th language
	5: 5th language
	6: 6th language
	7: 7th language
	8: 8th 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.

7.6.5 Non-volatile memory functions

7.6.5.1 Functions for controlling non-access-controlled areas

The following functions store the subscriber information (e.g. geographic, game scores, etc.) used in a program in a small capacity non-volatile memory (e.g. NVRAM) of the receiver. The stored information is used at the next broadcasting of the same event or a different event. Therefore, they do not perform general file input and output operations.

A group of information (e.g. information used by a single program) is recorded in a non-volatile memory with a name. In the following definitions of functions, the storage unit of this information is called “file” and the name of this information is called “file name.” If a file of the same name exists, the existing file is overwritten with the new file. That is, the new file is not appended to the existing file.

- writePersistentString(): Writes a character string in a non-volatile memory.

Syntax

```
Number writePersistentString(
    input String filename,
    input String buf
    [, input Date period]
)
```

Arguments

filename	File name
buf	Character string
period	Hold period

Return values

Number of bytes of character string written:	Success
NaN:	Failure

Description

This function writes a character string specified by buf in a file specified by filename. If a file of the same name exists, the existing file is overwritten with the string. The description of filename conforms to the namespace conventions defined in Chapter 9.

If a hold period, period is specified, the stored information is effective until the specified time. It cannot be read after the time. If period is not specified, no hold period is set.

- writePersistentNumber(): Write numeric data in a non-volatile memory.

Syntax

```
Number writePersistentNumber(
    input String filename,
    input Number data
    [, input Date period]
)
```

Arguments

filename	File name
data	Numeric value
period	Hold period

Return values

Number of bytes of character string written:	Success
NaN:	Failure

Description

This function writes a numeric value specified by data in a file specified by filename. If a file of the same name exists, the existing file is overwritten with the value. The description of filename conforms to the namespace conventions defined in Chapter 9.

If a hold period, the period argument is specified, the stored information is effective until the specified time. It cannot be read after the time. If the period argument is not specified, no hold period is set.

- writePersistentArray(): Writes the content of an array in a non-volatile memory.

Syntax

```
Number writePersistentArray(
    input String filename,
    input String structure,
    input Array data
    [, input Date period]
)
```

Arguments

filename	Name of a file that exists in a persistent storage device.
structure	Type specification of each element in a array

data	Array to be stored
period	Hold period

Return values

Number of bytes of the witten character string:	Success
NaN:	Failure

Description

This function is applicable to the file designated in filename. The array is written based on the type specified for each element designated by structure. The type of the array to be stored is based on the format in BinaryTable defined Section 7.5.2.2 However, it does not write any record length. The structure and type Arguments are defined as follows:

```
structure ::= field ["," field]*
type ::= "B" | "U" | "I" | "S" | "P"
```

The description of filename conforms to the namespace conventions defined in Chapter 9.

If a hold period, period is specified, the stored information is effective until the specified time. It cannot be read after the time. If the period argument is not specified, no hold period is set.

- readPersistentString(): Obtains the content of a file that exists in a non-volatile memory as a character string.

Syntax

```
String readPersistentString(input String filename)
```

Argument

filename	Name of a file that exists in a persistent storage device.
----------	--

Return values

Character string read:	Success
null:	Failure

Description

This function reads the character string was written by writePersistentString() from a file specified by filename, and returns the string. The description of filename conforms to the namespace conventions defined in Chapter 9.

- readPersistentNumber(): Obtains the content of a file that exists in a non-volatile memory as a numeric value.

Syntax

```
Number readPersistentNumber(input String filename)
```

Argument

filename	Name of a file that exists in a non-volatile memory.
----------	--

Return values

Numeric value read:	Success
NaN:	Failure

Description

Reads the numeric value that was written by writePersistentNumber() from a file specified by filename, and returns the value. The description of filename conforms to the namespace conventions defined in Chapter 9.

- readPersistentArray(): Obtains the content of a file that exists in a non-volatile memory as an array.

Syntax

```
Array readPersistentArray(  
    input String filename,  
    input String structure  
)
```

Arguments

filename	Name of a file that exists in a non-volatile memory.
structure	Type specification of each element in a array

Return values

Array that stores the obtained values:	Success
null:	Failure

Description

This function reads the array that was written by writePersistenArray() based on the type specification for each element specified by structure from a file specified by filename. The array is returned as the return value. The specification of type is based on the format in BinaryTable defined in Section 7.6.5. The description of filename conforms to the conventions on namespace defined in Chapter 9.

The value 'null' is returned when the file size is smaller than the record length specified by the structure argument.

- copyPersistent(): Copies a file that exists in a non-volatile memory.

Syntax

```
Number copyPersistent(  
    input String srcUri,  
    input String dstUri  
)
```

Arguments

srcUri	File name to be copied
dstUri	File name to store the copied file

Return values

Size of the file copied (byte number):	Success
NaN:	Failure

Description

This function copies the content of the file specified by srcUri to the file specified by dstUri. If data exists in the file specified by dstUri, it is overwritten with the content of the file specified by srcUri. The description of srcUri and dstUri conforms to the conventions on namespace defined in Chapter 9.

- getPersistentInfoList(): Obtains a list of files that exist in a non-volatile memory.

Syntax

```
Array getPersitentInfoList(input String type)
```

Arguments

type URI that indicates non-volatile memory or an area in it.

Return values

Array that stores file names: Success

null: Failure

Description

This function returns an array that stores a list of files that exist in a non-volatile memory specified by type. When no files exist, an array of length 0 is returned. The specification of a location specified by type conforms to the conventions on namespace defined in Chapter 9.

- deletePersistent(): Deletes a specified file from a specified type of a persistent storage.

Syntax

Number deletePersistent(input String filename)

Argument

filename URI that indicates the file name.

Return values

Size of the file that was deleted (byte number): Success

NaN: Failure

Description

This function deletes the file specified by filename. The description of filename conforms to the conventions on namespace defined in Chapter 9.

- getFreeSpace(): Returns the size of a free space in a specified type of a non-volatile memory.

Syntax

Number getFreeSpace(input String type)

Argument

type URI that indicates non-volatile memory or an area in it.

Return values

Non-negative integer that indicates the size of the free area (byte number): Success

NaN: Failure

Description

This function obtains the size of the free area in bytes , which is in a non-volatile memory specified in type.

7.6.5.2 Functions for controlling access-controlled areas

- setAccessInfoOfPersistentArray() : Set the access control information for a non-volatile memory.

Syntax

Number setAccessInfoOfPersistentArray(
 input String filename,
 input Number permissionType,

input Array permissionData

)

Arguments

filename	File name
permissionType	a numeric value representing the access control type
permissionData	Access control information

Return values

1:	Success
NaN:	Failure

Description

This function sets the access control information specified in permissionType and permissionData for the file specified in filename. The description of filename complies with the namespace conventions defined in Chapter 9.

The semantics of a value specified in permissionType is shown below.

Value	Semantics
1	Allowed to read/write only the content transmitted by selected services
2 ~ 127	reserved
128 ~ 255	Defined by a broadcaster

When a numeric value other than the values shown above is specified, or the receiver does not respond to a value in the range shown above, NaN is returned.

The content of permissionData is shown below.

Value in permissionType	Content of permissionData
1	First element : a network ID (original_network_id , a hexadecimal string in “0xXXXX” format) Second element : a transport stream ID (transport_stream_id, a hexadecimal string in “0xXXXX” format) Third element : a service ID (service_id, a hexadecimal string in “0xXXXX” format) When -1 is specified in the three elements, the access control information is applicable to all services.
2 ~ 127	reserved
128 ~ 255	Defined by a broadcaster

When executing this function resulted in modifying the access control information applicable to the file specified in filename, the content of the file specified in filename is destroyed.

- checkAccessInfoOfPersistentArray() : Verifies whether or not a non-volatile memory is accessible.

Syntax

```
Number checkAccessInfoOfPersistentArray(
    input String filename
)
```

Argument

filename	File name
----------	-----------

Return values

2:	Allowed to read/write
1:	Allowed to read only
0:	Not allowed to read/write
NaN:	Failure

Description

This function verifies whether or not the file specified in filename is accessible based on the access control information. When the file is readable and writable, 2 is returned. When the file is only readable, 1 is returned. When the file is unreadable and unwritable, 0 is returned.

The description of filename complies with the namespace conventions defined in Chapter 9.

- writePersistentArrayWithAccessCheck (): Writes the content of an array in a non-volatile memory.

Syntax

```
Number writePersistentArrayWithAccessCheck(
    input String filename,
    input String structure,
    input Array data
    [, input Date period]
)
```

Arguments

filename	Name of a file that exists in a non-volatile memory.
structure	Type specification of each element in a array
data	Array to be stored
period	Hold period

Return values

Number of bytes of the written character string:	Success
NaN:	Failure

Description

This function is applicable to the file designated by filename as long as the file is explicitly specified as writable. In this case, this function writes the array based on the type specified for each element in structure, and returns the error code as a return value. The type of the array to be stored is based on the format defined in BinaryTable of Section 7.5.2.2. However, it does not write any record length. The structure and type arguments are defined as follows:

```
structure ::= field [,field]*
type ::= "B" | "U" | "I" | "S" | "P"
```

The description of filename conforms to the namespace conventions defined in Chapter 9.

If a hold period, period is specified, the stored information is effective until the specified time. It cannot be read after the time. If the period argument is not specified, no hold period is set.

- readPersistentArrayWithAccessCheck(): Obtains the content of an file in a non-volatile memory as an array.

Syntax

```
Array readPersistentArrayWithAccessCheck(
    input String filename,
    input String structure
)
```

Arguments

filename	Name of a file that exists in a non-volatile memory.
structure	Type specification of each element in a array

Return values

Array containing the obtained values: Success



Failure

Description

This function is applicable to the file designated by filename as long as the file is explicitly specified as readable. In this case, this function reads the array that has been written with writePersistentArrayWithAccessCheck() based on the type specified for each element in structure, and returns the array as a return value. The description of filename conforms to the namespace conventions defined in Chapter 9. When the file is smaller than the record length specified in , the value 'null' is returned.

7.6.6 Extended APIs for Storing

7.6.6.1 Directory Management Functions

- saveDirAs() : Copies the structure of a directory on a storage device to other directory in the storage device.

Syntax

```
Number saveDirAs(
    input String src_path,
    input String dest_path
    [,input String dest_name]
)
```

Arguments

src_path	URI representing the directory to be copied
dest_path	URI representing the directory to which the structure is copied
dest_name	Name of the directory to which the structure is copied

Return values

1 :	Success
-1 :	src_path is invalid or the URI specified in src_path does not exist
-2 :	dest_path is invalid or the URI specified in dest_path does not exist
-3 :	Security breach regarding the file access (e.g. the file is specified as uncopiable based on the concerned digital-copy-control information)

NaN : Failure due to other causes

Description

The directory specified in `src_path` and the directories and files under it are copied to a location under the directory specified in `dest_path`. When the specified location is the existing directory, it is overwritten.

When `dest_name` is set, it is used as the name of the destination directory.

An error occurred when the directory specified in `dest_path` is under or over the directory specified in `src_path`.

A successful copy requires the concerned files and directories to retain not only valid content but also valid access control information including validity terms, copyability/uncopyability, and others. A security breach occurs when not less than one directory/file under the directory specified in `src_path` is uncopiable, no directory/file is copied.

- `saveDir()` : Copies the structure of a directory on a storage device to a location on the storage device based on the configuration of a receiver.

Syntax

```
String saveDir(
    input String src_path
    [, input String content_title
    [, input String content_type
    [, input String drive_type
    [, input Number drive_number]]]]
)
```

Arguments

<code>src_path</code>	URI representing the directory to be copied
<code>content_title</code>	Content title used for listing contents
<code>content_type</code>	Media type of the content
<code>drive_type</code>	Type of the drive
<code>drive_number</code>	Integer representing the drive (0 or larger)

Return values

String representing the destination path :	Success
null :	Failure (<code>src_path</code> is invalid, <code>src_path</code> does not exist, or the file access is a security breach)

Description

The directory specified in `src_path` and the directories and files under it are copied to an area, whose type is specified in `content_type` (When `content_type` is omitted, the destination area is the directory specified in a receiver's configuration. When `content_type` is available, the destination area is the directory whose type is specified a receiver's configuration.), of a drive, whose type is specified in `drive_type` (when `drive_type` is omitted, the type is specified based on a receiver's configuration). When the specified location is the existing directory, a new directory is created with a unique name that does not conflict with any existing directory name to store the copied information.

The available values of `drive_type` are:

"InternalHDD" :	Internal hard disk drive
"MemoryCard" :	Memory card
"ExternalDevice" :	External device

When drive_type is omitted, the default drive number specified in a receiver's configuration is used.

A successful copy requires the concerned files and directories to retain not only valid content but also valid access control information including validity terms, copiability/uncopiability, and others. A security breach occurs when not less than one directory/file under the directory specified in src_path is uncopiable, no directory/file is copied.

If content_title is omitted and the copied file retains the title information, the string retained as the title is used as the title of the destination file.

If content_title is omitted and the copied file does not retain the title information, no title information is retained by the destination file. Note that this does not prohibit a receiver from using other string as an alternative title to present the title information in a contents list.

- createDir() : Creates a directory.

Syntax

```
Number createDir(  
    input String path,  
    input String dir_name  
)
```

Arguments

path	URI representing the directory (a parent directory) to which the directory to be created belongs.
dir_name	Name of the directory to be created (a child directory).

Return values

1 :	Success
-1 :	path is invalid or path does not exist
-2 :	dir_name is invalid or dir_name exists
NaN:	Failure due to other causes

Description

This object creates a directory as a child directory of the parent directory specified in path.

- getParentDirName() : Obtains a parent directory name.

Syntax

```
String getParentDirName(input String path)
```

Argument

path	URI representing a directory or file
------	--------------------------------------

Return values

Path to the parent directory of the specified file/directory:	Success
null :	Failure (path is invalid, path does not exist, or other causes)

Description

This function returns the path of the parent directory located over the specified file or directory.

- getDirNames(): Obtains a directory name as an array.

Syntax

Array getDirNames(input String path)

Argument

path URI representing a directory (a parent directory)

Return values

Array of the string representing the child directory name: Success

null : Failure (path is invalid or path does not exist)

Description

This function returns the array of the strings representing the names of the (child) directories located under the specified (parent) directory. When no (child) directories exist under the specified (parent) directory, an array of length 0 is returned. When an array of length 2 or greater is returned, the order in which the names are placed in the array depends on an implementation.

- isDirExisting(): Verifies whether or not the specified directory exists.

Syntax

Boolean isDirExisting(input String path)

Argument

path URI representing a directory to be specified

Return values

true : The directory exists.

false : The directory does not exist.

Description

This function verifies whether or not the directory specified in path exists.

7.6.6.2 File Management Functions

- saveFileAs() : Copies a file on a storage device to the specified path in the storage device.

Syntax

```
Number saveFileAs(
    input String src_path,
    input String dest_path
)
```

Arguments

src_path URI representing the file to be copied

dest_path URI representing the file or directory to which the copied file is copied

Return values

1 :	Success
-1 :	src_path is invalid or the URI specified in src_path does not exist
-2 :	dest_path is invalid
-3 :	Security breach regarding the file access (e.g. the file is specified as uncopiable based on the concerned digital-copy-control information)
NaN :	Failure due to other causes

Description

The file specified in src_path is copied to a file (or a location under a directory) specified in dest_path.

A successful copy requires the concerned file to retain not only valid content but also valid access control information including validity terms, copiablity/uncopiability, and others. A security breach error occurs when the file specified in src_path is uncopiable.

- saveFile () : Copies a file on a storage device to the location in the storage device as specified in the receiver's configuration.

Syntax

```
String saveFile(  
    input String src_path  
    [, input String content_title  
    [, input String content_type  
    [, input String drive_type  
    [, input Number drive_number]]]]  
)
```

Arguments

src_path	URI representing the file to be copied
content_title	Content title used for listing contents
content_type	Media type of the content
drive_type	Type of the drive
drive_number	Integer representing the drive (0 or larger)

Return values

String representing the destination path :	Success
null :	Failure (src_path is invalid, src_path does not exist, or the file access is a security breach, or other causes)

Description

The file specified in src_path is copied to an area whose type is specified in content_type (When content_type is omitted, the destination area is the directory specified in a receiver's configuration. When content_type is available, the destination area is the directory whose type is specified a receiver's configuration.), of a drive, whose type is specified in drive_type (when drive_type is omitted, the type is specified based on a receiver's configuration).

When the specified location is the existing directory, a new file is created with a unique name that does not conflict with any existing directory name to store the copied information.

The available values of drive_type are:

"InternalHDD" :	Internal hard disk drive
"MemoryCard" :	Memory card
"ExternalDevice" :	External device

When drive_type is omitted, the default drive number specified in a receiver's configuration is used.

A successful copy requires the concerned file to retain not only valid content but also valid access control information including validity terms, copiablity/uncopiability, and others. An error occurs and no file is copied when the file specified in src_path is uncopiable.

If content_title is omitted and the copied file retains the title information, the string retained as the title is used as the title of the destination file.

If content_title is omitted and the copied file does not retain the title information, no title information is retained by the destination file. Note that this does not prohibit a receiver from using other string as an alternative title to present the title information in a contents list.

- getFileNames(): Obtains all files names located under the specified directory.


Syntax

Array getFileNames(input String path)


Argument

path URI representing a directory

Return values

String representing the file names:	Success
null :	Failure (path is invalid, path does not exist) 

Description

This function returns the array of the string representing the names of the files located under the specified directory.  When no files exist in the specifed directory, an array of length 0 is returned. When an array of length 2 or greater is returned, the order in which the names are placed in the array depends on the receiver.

- isFileExisting(): Verifies whether or not the specified file exists.

Syntax

Boolean isFileExisting(input String path)

Argument

path URI representing a file to be specified

Return values

true :	The file exists.
false :	The file does not exist.

Description

This function verifies whether or not the file specified in path exists.

7.6.6.3 File Input/Output Functions

- writeArray(): Writes the content of an array in a file on a storage device.

Syntax

```
Number writeArray(  
    input String filename,  
    input String structure,  
    input Array data  
    [, input Date period]  
)
```

Arguments

filename	URI representing a file name
structure	Type specification of each element in a array
data	Array to be stored
period	Hold period

Return values

Number of bytes of the written character string:	Success
-1:	Failure because the storage device is busy in writing
-2:	Failure because there is not enough capacity on the storage device
NaN:	Failure due to other causes

Description

This function is applicable to the file designated in filename. The array is written based on the type specified for each element designated in structure, and the error code is returned as a return value. The type of the array to be stored is based on the format in BinaryTable defined in Section 7.5.2.2. However, it does not write any record length. The structure and type arguments are defined as follows:

```
structure ::= field[" , " field]*  
type ::= "B" | "U" | "I" | "S" | "P"
```

The description of filename conforms to the namespace conventions defined in Chapter 9.

If a hold period, period is specified, the stored information is effective until the specified time. It cannot be read after the time. If the period argument is not specified, no hold period is set.

(See the writePersistentArray() description in Section 7.6.5.)

- readArray(): Obtains the content of a file that exists in a storage device.

Syntax

```
Array readArray(  
    input String filename,  
    input String structure  
)
```

Arguments

filename	URI representing a file name.
structure	Type specification of each element in a array

Return values

Array that stores the obtained values:	Success
null:	Failure

Description

This function reads the array that was written by writeArray() based on the type specification for each element specified by structure from a file specified by filename. The array is returned as the return value. The specification of structure is based on the format in the structure argument of writeArray() defined in Section 7.6.6.

The description of filename conforms to the conventions on namespace defined in Chapter 9.

The value 'null' is returned when the file size is smaller than the record length specified by the structure argument.

(See the readPersistentArray() description in Section 7.6.5.)

7.6.6.4 Inquiry Functions

- getDirInfo(): Obtains a directory information.

Syntax

```
Array getDirInfo(
    input String path
    [, input String additionalinfo]+
)
```

Arguments

path	URI representing a directory to be specified
additionalinfo	String (case-sensitive) identifying the type of additional information about the directory. A set of strings identifying a type of additional information is defined in an operational standard regulation.

Return values

Array that stores the directory information:	Success
Array[0] :	Expiration date (Date)or null
Array[1] and following Arrays :	If the information specified in additionalinfo is obtained, true is returned. If the specified information is not obtained, false is returned. However, a string unknown to the implemented system is obtained, false is returned.
null:	Failure (path is invalid, path does not exist, or other causes)

Description

This function obtains information about the specified file including the expiration date and attributes specified in an operational standard regulation. When the three conditions are satisfied, the expiration date and time is returned: the specified file is a directory that has been generated upon storing the content transmitted with the data carousel transmission specification, the

directory has been mapped to a module responsible for the transmission, and the module has been specified with the expire descriptor. If no expiration date is specified, null is returned in Array [0].

- getFileInfo(): Obtains a file information.

Syntax

```
Array getFileInfo(
    input String path
    [, input String additionalinfo]+
)
```

Arguments

path	URI representing a file to be specified
additionalinfo	String (case-sensitive) identifying the type of additional information about the file. The available strings are:
"copiable":	The file is allowed to be copied to make any generation of the file.
Other strings:	Other strings that identify the type of additional information are defined in an operational standard regulation.

Return values

Array that stores the directory information:	Success
Array[0]:	File size in bytes
Array[1]:	Update date and time (Date)
Array[2]:	Expiration date and time (Date)
Array[3] and following Arrays:	If the information specified in additionalinfo is obtained, true is returned. If the specified information is not obtained, false is returned. However, a string unknown to the implemented system is obtained, false is returned.
null:	Failure (path is invalid, path does not exist, or other causes)

Description

This function obtains information about the specified file including the expiration date, the update date, copiablity/uncopiability and attributes specified in an operational standard regulation.

The returned updated date and time is the date and time when the directory was updated by a receiver during storing data service, or with the saveXXXX() function or the createDir() function.

When the three conditions are satisfied, the expiration date and time is returned: the specified file is a file that has been generated upon storing the content transmitted with the data carousel transmission specification, and the module has been specified with the expire descriptor.

- getContentSource(): Verifies whether the currently executed content was invoked from a broadcasting service or invoked from a storage device.

Syntax

```
Boolean getContentSource(input String source)
```

Argument

source String identifying the environment from which the currently executed BML document was invoked. The available strings are:

"onair" : Broadcasting service
"storage" : File on a storage device
"partialTS" : Partial TS
"HTTPServer" : Communication line connected via IP

Return values

true: success
false: failure

Description

When source is "onair" and the currently executed BML document was invoked from a broadcasting service, true is returned.

When source is "storage" and the currently executed BML document was invoked from a storing storage, true is returned.

When source is "partialTS" and the currently executed BML document was invoked from Partial TS recorded in DVHS and others, true is returned.

When source is "HTTPServer" and the currently executed BML document was obtained using http via a communication line connected via IP, true is returned.

In other cases than the above four cases, false is returned. However, an additional string available to source, which may be defined in future extended definitions/operational standard regulations, may add the case where true is returned.

- getStorageInfo(): Obtains a storage device information.

Syntax

```
Array getStorageInfo(
    input String drive_type
    [, input Number drive_number]
    [, input String additionalinfo]+
)
```

Arguments

drive_type Type of the drive
drive_number Integer representing the drive number (0 or larger)
additionalinfo String (case-sensitive) identifying the information about the storage device. Available strings are defined in an operational standard regulation.

Return values

Array representing the storage device information: Success

Array[0] : Total drive capacity in 1024-byte units
Array[1] : Free capacity per drive type in 1024-byte units
Array[2] and following Arrays : If the information specified in additionalinfo is obtained, true is returned. If the specified information is not obtained, false is returned. However, a string

unknown to the implemented system is obtained,
false is returned.

null: Failure (the specified drive type is invalid, or
the specified type of drive does not exist)

Description

This function obtains the capacity of the storage device whose drive type/drive number is specified and other additional information as defined in an operational standard regulation. The available string identifying the drive type are:

"InternalHDD" : Internal hard disk drive

"MemoryCard" : Memory card

"ExternalDevice" : External device

When the drive number is omitted, the default drive number defined in a receiver's configuration is used.

- getCarouselInfo(): Obtains a carousel information.

Syntax

```
Array getCarouselInfo(
    input String stream_ref
    [, input String additionalinfo])+
)
```

Arguments

stream_ref	URI identifying a component stream
additionalinfo	String (case-sensitive) identifying the type of additional information about the carousel. The available strings are:
"copiable" :	The file is allowed to be copied to make any generation of the carousel.
Other strings:	Other strings that identify the type of additional information are defined in an operational standard regulation.

Return values

Array representing the carousel information:	Success
Array[0] :	Carousel size in bytes
Array[1] :	Number of modules
Array[2] and following Arrays :	If the information specified in additionalinfo is obtained, true is returned. If the specified information is not obtained, false is returned. However, a string unknown to the implemented system is obtained, false is returned.
null:	Failure (stream_ref is invalid, the carousel specified in stream_ref does not exist, or other causes)

Description

This function obtains information about the carousel transmitted in the specified component including the total size of the modules, the number of the modules, copiability/uncopiability and other additional information as defined in an operational standard regulation.

- getModuleInfo(): Obtains a module information.

Syntax

```
Array getModuleInfo(
    input String module_ref
    [, input String additionalinfo]+
)
```

Arguments

module_ref	URI identifying a module in the data carousel
additionalinfo	String (case-sensitive) identifying the type of additional information about the carousel. The available strings are:
"copiable":	The file is allowed to be copied to make any generation.
Other strings:	Other strings that identify the type of additional information are defined in an operational standard regulation.

Return values

Array representing the module information:	Success
Array[0] :	Module size in bytes
Array[1] :	Media type (String)
Array[2] :	Expiration date (Date)
Array[3] and following Arrays :	If the information specified in additionalinfo is obtained, true is returned. If the specified information is not obtained, false is returned. However, a string unknown to the implemented system is obtained, false is returned.
	Note that Array [3] and the following Arrays are reserved for future extensions.
null:	Failure (module_ref is invalid, the module specified in module_ref does not exist, or other causes)

Description

This function obtains information about the module transmitted in the specified component including the size, the media type, the expiration date, copiability/uncopiability and other additional information as defined in an operational standard regulation.

7.6.6.5 Data Carousel Storage Functions

- saveCarouselAs() : Copies an entire carousel in a specified path on a storage device.

Syntax

```
Number saveCarouselAs(
    input String src_component,
    input String dest_path
)
```

Arguments

src_component	URI representing the stream component of the carousel to be copied
dest_path	URI representing the file on the storage device, to which the carousel is copied

Return values

1 :	Success
-1 :	src_component is invalid or the carousel specified in src_component does not exist
-2 :	dest_path is invalid
-3 :	Security breach regarding the file access (e.g. the file is specified as uncopiable based on the concerned digital-copy-control information)
NaN :	Failure due to other causes

Description

The modules of the carousel transmitted in the component specified in src_component are copied to the directory specified in dest_path. Note that the resources contained in the modules are also copied. When the specified location is the existing directory, it is overwritten.

A successful copy requires the concerned modules and resources to retain not only valid content but also valid access control information including validity terms, copiability/uncopiability, and others. If the specified carousel is not allowed to be copied, this operation causes security breach error.

- saveCarousel() : Copies an entire carousel in an area on a storage device based on a receiver's configuration.

Syntax

```
String saveCarousel(
    input String src_component
    [, input String content_title
    [, input String content_type
    [, input String drive_type
    [, input Number drive_number]]]]
)
```

Arguments

src_component	URI representing the stream component of the carousel to be copied
content_title	Content title used for listing contents
content_type	Media type of the content
drive_type	Type of the drive
drive_number	Integer representing the drive number (0 or larger)

Return values

String representing the destination path :	Success
null :	Failure (src_path is invalid, src_path does not exist, or the file access is a security breach)

Description

The modules and the resources of the carousel transmitted in the component specified in `src_component` are stored in an area, whose type is specified in `content_type` (When `content_type` is omitted, the destination area is the directory specified in a receiver's configuration. When `content_type` is available, the destination area is the directory whose type is specified a receiver's configuration.), of a drive, whose type is specified in `drive_type` (when `drive_type` is omitted, the type is specified based on a receiver's configuration). When the specified area already exists, the receiver assigns a unique name that does not conflict with any existing directory name to a newly created area to store the copied information.

The available values of `drive_type` are:

"InternalHDD" : Internal hard disk drive
"MemoryCard" : Memory card
"ExternalDevice" : External device

When `drive_number` is omitted, the default drive number specified in a receiver's configuration is used.

A successful copy requires the concerned modules and resources to retain not only valid content but also valid access control information including validity terms, copiability/uncopiability, and others. An error occurs when the specified carousel is uncopiable. In this case, no module/resource is copied.

If `content_title` is omitted and the copied file retains the title information, the string retained as the title is used as the title of the destination file.

If `content_title` is omitted and the copied file does not retain the title information, no title information is retained by the destination file. Note that this does not prohibit a receiver from using other string as an alternative title to present the title information in a contents list.

- `saveModuleAs()` : Copies a module in a specified path on a storage device.

Syntax

```
Number saveModuleAs(
    input String src_module,
    input String dest_path
)
```

Arguments

<code>src_module</code>	URI representing the module of the carousel to be copied
<code>dest_path</code>	URI representing the file on the storage device, to which the module is copied

Return values

1 :	Success
-1 :	<code>src_module</code> is invalid or the module specified in <code>src_module</code> does not exist
-2 :	<code>dest_path</code> is invalid
-3 :	Security breach regarding the file access (e.g. the file is specified as uncopiable based on the concerned digital-copy-control information)
NaN :	Failure due to other causes

Description

A module transmitted in a component specified with `src_component` is copied to a directory specified with `dest_path`.

A successful copy requires the concerned modules and resources to retain not only valid content but also valid access control information including validity terms, copiability/uncopiability, and others. If the specified carousel is not allowed to be copied, this operation causes security breach error.

- `saveModule()` : Copies a module in an area on a storage device based on a receiver's configuration.

Syntax

```
String saveModule(  
    input String src_module  
    [, input String content_title  
    [, input String content_type  
    [, input String drive_type  
    [, input Number drive_number]]]]  
)
```

Arguments

<code>src_module</code>	URI representing the module to be copied
<code>content_title</code>	Content title used for listing contents
<code>content_type</code>	Media type of the content
<code>drive_type</code>	Type of the drive
<code>drive_number</code>	Integer representing the drive number (0 or larger)

Return values

String representing the destination path :	Success
null :	Failure (<code>src_module</code> is invalid, <code>src_module</code> does not exist, or the file access is a security breach)

Description

A module transmitted in a component specified with `src_component` is stored in an area, whose type is specified in `content_type` (When `content_type` is omitted, the destination area is the directory specified in a receiver's configuration. When `content_type` is available, the destination area is the directory whose type is specified a receiver's configuration.), of a drive, whose type is specified in `drive_type` (when `drive_type` is omitted, the type is specified based on a receiver's configuration). When the specified area already exists, the receiver assigns a unique name that does not conflict with any existing directory name to a newly created area to store the copied information.

The available values of `drive_type` are:

"InternalHDD" :	Internal hard disk drive
"MemoryCard" :	Memory card
"ExternalDevice" :	External device

When `drive_number` is omitted, the default drive number specified in a receiver's configuration is used.

A successful copy requires the concerned directory and files to retain not only valid content but also valid access control information including validity terms, copiability/uncopiability, and

others. An error occurs when the specified module is uncopiable. In this case, no module/resource is copied.

If content_title is omitted and the copied module retains the title information specified with Title descriptor, the string retained as the title is used as the title of the destination file.

If content_title is omitted and the copied file does not retain any available title information, no title information is retained by the destination file. Note that this does not prohibit a receiver from using other string as an alternative title to present the title information in a contents list.

- saveResourceAs() : Copies a resource in a module to a specified path on a storage device.

Syntax

```
Number saveResourceAs(
    input String src_resource,
    input String dest_path
)
```

Arguments

src_resource	URI representing the resource to be copied
dest_path	URI representing the file on the storage device, to which the module is copied

Return values

1 :	Success
-1 :	src_resource is invalid or the module specified in src_resource does not exist
-2 :	dest_path is invalid
-3 :	Security breach regarding the file access (e.g. the file is specified as uncopiable based on the concerned digital-copy-control information)
NaN :	Failure due to other causes

Description

The resource specified in src_resource is copied to the file (or directory) specified in dest_path.

A successful copy requires the concerned file to retain not only valid content but also valid access control information including validity terms, copiability/uncopiability, and others. If the specified carousel is not allowed to be copied, this operation causes security breach error.

- saveResource() : Copies a resource in a module to an area on a storage device based on a receiver's configuration.

Syntax

```
String saveResource(
    input String src_resource
    [, input String content_title
    [, input String content_type
    [, input String drive_type
    [, input Number drive_number]]]]
)
```

Arguments

src_resource	URI representing the resource to be copied
content_title	Content title used for listing contents
content_type	Media type of the content
drive_type	Type of the drive
drive_number	Integer representing the drive number (0 or larger)

Return values

String representing the destination path:	Success
null:	Failure (src_resource is invalid, src_resource does not exist, or the file access is a security breach)

Description

The resource specified in src_resource is stored in an area, whose type is specified in content_type (When content_type is omitted, the destination area is the directory specified in a receiver's configuration. When content_type is available, the destination area is the directory whose type is specified a receiver's configuration.), of a drive, whose type is specified in drive_type (when drive_type is omitted, the type is specified based on a receiver's configuration). When the specified area already exists, the receiver assigns a unique name that does not conflict with any existing file name to a newly created area to store the copied information.

The available values of drive_type are:

"InternalHDD" :	Internal hard disk drive
"MemoryCard" :	Memory card
"ExternalDevice" :	External device

When drive_number is omitted, the default drive number specified in a receiver's configuration is used.

A successful copy requires the concerned resource to retain not only valid content but also valid access control information including validity terms, copiability/uncopiability, and others. An error occurs when the specified resource is uncopiable. In this case, no information is copied.

If content_title is omitted and the copied resource retains the title information specified with Title descriptor, the string retained as the title is used as the title of the destination file.

If content_title is omitted and the copied resource does not retain any available title information, no title information is retained by the destination file. Note that this does not prohibit a receiver from using other string as an alternative title to present the title information in a contents list.

7.6.7 Interaction Channel functions

7.6.7.1 Communication Functions assuming simple protocols including BASIC procedures

- connect(): Establishes a connection.

Syntax

```
Number connect (  
    input String tel,  
    [input String hostNo,]
```

input Boolean bProvider,
input Number speed,
input Number timeout
)

Arguments

tel	Telephone number to call
hostNo	Host number (required only when X.28 Protocol is used)
bProvider	Network specify flag
speed	Connection line speed (1: Low, 2: High)
timeout	Specifies a period of time (in milliseconds). When the specified period expires, the concerned process is assumed to be timed-out.

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
NaN:	Failure by other causes

Description

This function dials a telephone number specified with tel and tries to establish a communication. When using X.28 Protocol, hostNo is specified. If bProvider (network specify flag) is true, a communication provider identification code preset in the receiver can be attached at the beginning of the telephone number. If no connection establishes within a time specified with timeout, the dialling fails.

If speed is set to 1, V.22bis+MNP4 negotiation is performed for PSTN and PDC, and PIAFS32K negotiation is performed for the PHS network. If speed is set to 2, the receiver tries for the highest possible speed.

- disconnect(): Disconnects the line.

Syntax

Number disconnect ()

Argument

None

Return values

1:	Success
NaN:	Failure

Description

This function disconnects the currently established communication line.

- sendBinaryData(): Sends binary data.

Syntax

```
Number sendBinaryData (  
    input String uri,  
    input Number timeout  
)
```

Arguments

uri	URI of the file that has data to send
timeout	Specifies a period of time (in milliseconds). When the specified period expires, the concerned process is assumed to be timed-out.

Return values

1:	Success
-1:	Parameter error
-2:	Line was disconnected during transfer.
-3:	Time-out occurred.
NaN:	Failure by other causes

Description

This function sends binary data stored in a file specified with uri through a communication line that has been established. If the function cannot send data within a time specified with timeout, it exits returning an error.

- receiveBinaryData(): Receives binary data.

Syntax

```
Number receiveBinaryData (  
    input String uri,  
    input Number timeout  
)
```

Arguments

uri	URI of a file that stores binary data
timeout	Specifies a period of time (in milliseconds). When the specified period expires, the concerned process is assumed to be timed-out.

Return values

1:	Success
-1:	Parameter error
-2:	Line was disconnected during transfer.
-3:	Time-out occurred.
NaN:	Failure by other causes

Description

This function receives the binary data in a file specified with uri through a communication line that has been established. If the function cannot receive the data within a time specified with timeout, it exits returning an error.

- sendTextData(): Sends text data.

Syntax

```
Number sendTextData (
    input String text,
    input Number timeout
)
```

Arguments

text	Text data to be sent
timeout	Specifies a period of time (in milliseconds). When the specified period expires, the concerned process is assumed to be timed-out.

Return values

1:	Success
-1:	Parameter error
-2:	Line was disconnected during transfer.
-3:	Time-out occurred.
NaN:	Failure by other causes

Description

This function sends the text data specified with text through a communication line that has been established. If the function cannot send the data within a time specified with timeout, it exits returning an error.

- receiveTextData(): Receives text data.

Syntax

```
String receiveTextData (input Number timeout)
```

Argument

timeout	Specifies a period of time (in milliseconds). When the specified period expires, the concerned process is assumed to be timed-out.
---------	--

Return values

The following information is returned depending on the success or failure of the operation.

Text received:	Success
null:	Failure

Description

This function receives text data through a communication line that has been established and returns it. If the function cannot receive text data within a time specified with timeout, it exits returning an error.

7.6.7.2 Delayed call functions assuming simple protocols including BASIC procedures

- registerTransmission(): Registers calling.

Syntax

```
Number registerTransmission (
    input Date start_time,
    input Date first_timelimit,
    input Date last_timelimit,
    input Number redial_interval,
    input String pgm_uri,
    input String regname,
    input String strdata+
)
```

Arguments

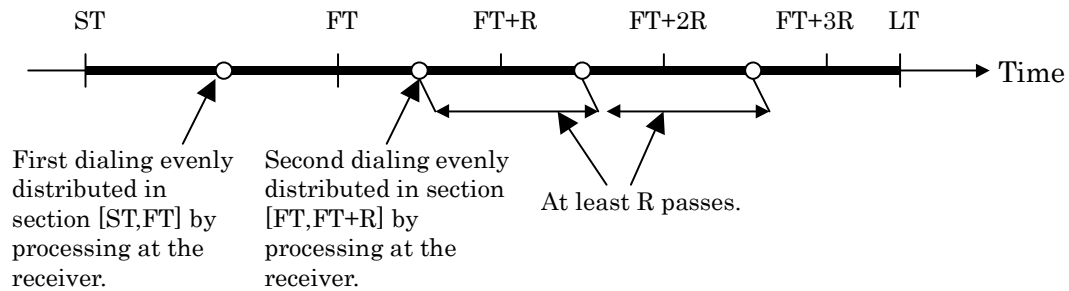
start_time	Start time
first_timelimit	First time limit
last_timelimit	Last time limit
redial_interval	Redial interval (in seconds)
pgm_uri	URI of text file that stores the delayed call function
regname	Registration name of delayed call
strdata	Character string(s) used as a message

Return values

Value of registration ID:	Success
-1:	Parameter error
-100:	Failure by exceeding time limit
-101:	Failure by exceeding number of registration
-102:	Failure by invalid content of pgm_uri
-103:	Failure by invalid number/length of strdata
NaN:	Abort by other causes

Description

This function registers the startConnection() call function that is stored in a URI specified with pgm_uri to be performed with the argument specified with strdata at times evenly distributed for each terminal in a section of time from start_time to first_timelimit. If a dialling failed due to the busy line before reaching first_timelimit, the second dialling is performed at a time evenly distributed in a section of time from first_timelimit to first_timelimit+redial_interval. The third or later dialling is performed after at least a time specified by redial_interval has passed (see Figure 7-1). Unless a connection has failed, dialling is performed only once.



Legend: ST: start_time, FT: first_timelimit, LT: last_timelimit, R: redial_interval

Figure 7-1 Scheduling of Dialling Time and Redialling Times by Receiver

The information to be registered to establish a connection consists of URI specified in `pgm_uri`, one or more `strdata` used as a message, `regname` to be displayed to human, `service_id`, `event_id`, and the current time. This information is recorded in an area on a terminal.

When this registration has successfully exits, the function returns a registration ID with which the registration is uniquely identified in the receiver.

- `registerTransmissionStatus()`: Registers a specified transmission status.

Syntax

Number `registerTransmissionStatus` (input Number status)

Argument

status	Status code
--------	-------------

Return values

1:	Success
-1:	Parameter error
-110:	Invoked with a context other than a dialing function
NaN:	Failure by other causes

Description

This function registers the status (e.g. “Done”, “Waiting for redialling”) specified with status of the currently executed delayed call function. The status is registered in the receiver. The status has one of the following values.

- 1: Done
- 2: Not dialled
- 3: Waiting for redialling
- 4: Failed due to abnormal line condition

- `getTransmissionStatus()`: Obtains a registered transmission status.

Syntax

Number `getTransmissionStatus` (input Number regid)

Arguments

regid	Registration ID
-------	-----------------

Return values

0:	No dialling with the specified registration ID is registered.
----	---

1:	Done
2:	Not dialled
3:	Waiting for redialling
4:	Failed due to an abnormal line condition
-1:	Parameter error
-110:	Invoked with a context other than a dialling function
NaN:	Failure by other causes

Description

This function obtains the registered transmission status (e.g. “Done” and “Waiting for redialling”) applicable to the delayed call specified in regid from the receiver.

- setDelayedTransmissionDataOverBASIC(): Registers a data transmission to be performed at a specified time using BASIC procedures and other simple protocol.

Syntax

```
Number setDelayedTransmissionDataOverBASIC(  
    input String regname,  
    input String tel,  
    [input String hostNo,]  
    input String bProvider,  
    input Date start_time,  
    input Date end_time,  
    input Number retry_interval,  
    input String strtext  
)
```

Arguments

regname	Registration name of the delayed call
tel	Telephone number to be called
hostNo	Host number (required only when X.28 Protocol is used)
bProvider	Network identification flag
start_time	Time to start the data transmission
end_time	Time to end the data transmission
retry_interval	Trial interval (in seconds)
strtext	Character string(s) used as a message

Return values

1:	Success
-1:	Parameter error
-100:	Failure by specifying a past time
-101:	Failure by exceeding number of registration
-102:	Failure in writing the data into a restricted area

NaN: Failure by other causes

Description

This function registers a data transmission to be performed at a specified time using BASIC procedures and other simple protocol. Once the data transmission information has been successfully registered, the specified message is transmitted at the time specified with `start_time` to a telephone number specified in `tel` using BASIC procedures and other simple protocols, When the X.28 protocol is used, `hostNo` is required.

When the `bProvider` network identification flag is true, a carrier identification code defined in a receiver's configuration is placed at the beginning of the called telephone number.

If a dialling at the time specified with `start_time` failed, the second dialling is performed when the period specified in `retry_interval` expires. The third or following dialling is performed when at least a period specified in `retry_interval` has passed since the last failure until the message is successfully sent and received. If the message has not been successfully sent and received before the time specified in `end_time`, the registered data transmission information is deleted and the failure is recorded as the result. In this case, no more retry is made.

Unless a connection has failed, dialling is performed only once.

Note that if the data is in transmission at the time specified in `end_time`, the transmission must be continued until it successfully exits.

The available values of an interval from `start_time` to `end_time`, text data size, time out for calling, and time out for sending are defined in an operational standard regulation.

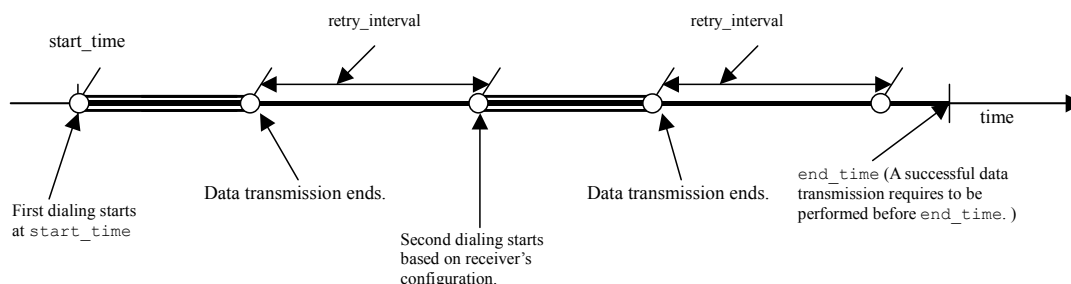


Figure 7-2 How setDelayedTransmissionDataOverBASIC() works

The `regname` argument is used to present the transmitted data to human operators. Program identifications including `service_id` and `event_id` should be registered in order to identify the registering program. A receiver should be capable to display and delete information registered with the `setDelayedTransmissionDataOverBASIC()` function.

7.6.7.3 Communication functions using the mass calls reception service

Note: The mass calls reception service is provided only in Japan. It is also called "Tele-Gong".

- `vote()`: Calls the Mass Calls Reception Service

Syntax

```
Number vote(
    input String tel,
    input Number timeout
)
```

Arguments

tel Telephone number character string

timeout Specifies a period of time (in milliseconds). When the specified period expires, the concerned process is assumed to be timed-out.

Return values

1:	Success
-1:	Parameter error
-4:	No dial tone detected
-6:	Disconnection enforced
-7:	Modem in use
NaN:	Failure by other causes

Description

This function calls the Mass Call Reception Service. It dials a number specified in tel and establishes a communication. Use the connect() function for a cut-through call.

7.6.7.4 Functions for encrypted communication using CAS

- startCASEncryption(): Declares the beginning of data encryption using CAS.

Syntax

```
Number startCASEncryption(  
    input Number provider,  
    input Number centerID  
)
```

Arguments

provider	Identifies a pay service operator
centerID	Center ID

Return values

1:	Success
-1:	Parameter error
-50:	Does not have CAS capability
-51:	IC card is not inserted
NaN:	Failure by other causes

Description

This function makes the CAS encryption function applicable to data communication. It must be performed before establishing a connection.

- endCASEncryption(): Declares the end of data encryption using CAS.

Syntax

```
Number endCASEncryption()
```

Argument

None

Return values

1:	Success
NaN:	Abnormal end

Description

This function quits applying CAS to data encryption

- transmitWithCASEncryption(): Transmits a message encrypted using CAS.

Syntax

```
Array transmitWithCASEncryption (
    input String sendData,
    input Number timeout
)
```

Arguments

sendData	Text data to be sent
timeout	Specifies a period of time (in milliseconds). When the specified period expires, the concerned process is assumed to be timed-out.

Return values

Array[0]:	Values representing result codes
1:	Success
-1:	Parameter error
-2:	Line was disconnected during transmission.
-3:	Time-out occurred.
-50:	Does not have CAS capability
-51:	IC card is not inserted.
NaN:	Failure by other causes
Array[1]:	Decoded, received text

Description

This function encrypts a message specified in sendData using CAS and sends it to the center through a communication line that has been established. It then receives a corresponding message from the center and decodes it using CAS. If the message transmission does not complete within a time specified by timeout, the function exits returning an error code.

7.6.7.5 Functions for communication with public key encryption not using CAS

- setEncryptionKey(): Sets a public key.

Syntax

```
Number setEncryptionKey(input Number key+)
```

Arguments

key1,...,keyN	Integers constituting a public key
---------------	------------------------------------

Return values

1:	Success
-1:	Parameter error
NaN:	Abnormal end by other causes

Description

This function sets a public key. The key length, the number of arguments, and data formats of arguments are defined in an operational standard regulation.

- beginEncryption(): Turns on an encryption.

Syntax

Number beginEncryption()

Argument

None

Return values

1:	Success
-40:	Key undefined
NaN:	Abnormal end by other causes

Description

This function starts an encryption of data transmitted through an interactive channel that has been established. The encryption algorithm to be used or others are defined in an operational standard regulation.

- endEncryption(): Ends an encryption.

Syntax

Number endEncryption()

Arguments

None

Return values

1:	Success
NaN:	Abnormal end

Description

This function ends an encryption of data transmitted through an interactive channel that has been established.

7.6.7.6 Communication functions assuming TCP/IP

A communication with TCP/IP is assumed to be connected to the two types of locations:

- 1) a location that has been configured on a receiver
- 2) a location that has been described in the concerned content as the concerned broadcaster's intended destination.

A connection with TCP/IP is assumed:

- 1) An IP packet request forces a receiver to establish a connection based on its configuration. In this case, the concerned content is not responsible for controlling

disconnections. However when a PPP connection has been established the connection can be disconnected as long as an end user explicitly verifies and allows the disconnection through a BML document.

- 2) Broadcasters' content request the receiver to establish connection based on either the receiver setting or description in the content itself. In this case, as a general rule, the concerned content is responsible for controlling disconnections.

- 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 NumberlineType]
)
```

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

1:	Normal end
-1:	Parameter error
-2:	Could not make an storage area available to store the configuration
-3:	The configuration is cancelled by an end user. (this value is used when an operational standard regulation requires a confirming message to be displayed on a receiver)
-4:	Could not configure due to an invalid service operator identification
NaN:	Abort by other causes

Description

This function is applicable to a terminal that has an IP connection feature. It stores connection parameters in a non-volatile memory. The connection parameters include the Internet service operator and related parameters that are specific to the data broadcasting program currently received. The configured parameters are effective until it is overwritten using this function or a receiver's feature. When the bProvider network identification flag is true, a carrier identification code defined in a receiver's configuration is placed at the beginning of the called telephone number.

Once the configured information has been successfully stored using this function, a receiver retains information to identify the service operator who have done the configuration and other related information. The retained information is used by the receiver to verify whether or not a writing access is allowed.

Actual usage of the identifying information and status is defined in an operational standard regulation.

For more information on values applicable to lineType, refer to the Return values list in the getConnection Type() section. When no value is set for lineType, a value configured in a receiver is recognized as a default value.

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

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 etISPParams() function is executed. Detailed usage of strings are defined in an operational standard regulation.
null:	Failure

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.

- 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 Mobile phone/PHS is preferred to be used and line types are detectable.)
-302:	External communication device was not available (When Mobile phone/PHS is preferred to be used 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. Note that no configured information stored in a receiver affect reference.

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/PHS 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 Mobile phone/PHS is preferred to be used and line types are detectable.)
-302:	External communication device was not available (When Mobile phone/PHS is preferred to be used and line types are detectable.)
NaN:	Failure by other causes

Description

This function establishes a PPP connection according to the receiver's 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/PHS 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 CellularSystem)
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 returned, when the preferred line type is PHS and the specific type (PIAFS2.0 or PIAFS2.1) is not identified. The return value 300 is returned, 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.

- saveHttpServerFileAs() : Copies a file on an HTTP server to the specified path in a storage device.

Syntax

```
Array saveHttpServerFileAs(
    input String src_path,
    input String dest_path
)
```

Arguments

src_path	URI representing the file on an HTTP server to be copied
dest_path	URI representing the path of the file to which the specified file is copied

Return values

Array value that contains information on success or failure of the operation

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
-602:	Invalid file name
-603:	Not enough storage space available on the storage device
-604:	Error during storing file
-700:	Service was disconnected

NaN: Failure by other causes

Array[1]: Status-Code in HTTP1.1
 Array[2]: Byte length of received file
 Array[3]: Content-Type in HTTP1.1 (Media type described in header)

Description

The file in an http server specified in src_path is copied to a file specified in dest_path in the storage_device.

A return value is an array value consisting of Status-Code in HTTP1.1, the byte length of the received file, and the media type. As the byte length of the received file, the value of Content-Length used by http is returned. When the file is not successfully stored, Array [2] contains 0.

- saveHttpServerFile() : Copies a file on an HTTP server to an area in a storage device as specified in a receiver's configuration.

Syntax

```
Array saveHttpServerFile(
    input String src_path
    [,input String content_title
    [,input String content_type
    [, input String drive_type
    [, input Number drive_number]]]]
)
```

Arguments

src_path	URI representing the resource to be copied
content_title	Content title used for listing contents
content_type	Media type of the content
drive_type	Type of the drive
drive_number	Integer representing the drive number (0 or larger)

Return values

Array value that contains information on success or failure of the operation

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
-601:	Invalid storage device was specified
-602:	Invalid file name was specified

- 603: Not enough storage space available on the storage device
- 604: Error during storing file
- 700: Service was disconnected
- NaN: Failure by other causes

Array[1]:	Status-Code in HTTP1.1
Array[2]:	Byte length of received file
Array[3]:	Content-Type in HTTP1.1 (Media type described in header)

Description

The directory on an http server specified in `src_path` is copied to an area, whose type is specified in `content_type` (When `content_type` is omitted, the destination area is the directory specified in a receiver's configuration. When `content_type` is available, the destination area is the directory whose type is specified in a receiver's configuration.), of a drive, whose type is specified in `drive_type` (when `drive_type` is omitted, the type is specified based on a receiver's configuration). When the specified location already exists, a unique name that does not conflict with any existing directory name is assigned to a newly created directory to store the copied information.

The available values of `drive_type` are:

- "InternalHDD" : Internal hard disk drive
- "MemoryCard" : Memory card
- "ExternalDevice" : External device

When `drive_number` is omitted, the default drive number specified in a receiver's configuration is used.

A return value is an array value consisting of Status-Code in HTTP1.1, the byte length of the received file, and the media type. As the byte length of the received file, the value of Content-Length used by http is returned. When the file is not successfully stored, Array [2] contains 0.

If `content_title` is omitted, no title information is retained by the destination file. Note that this does not prohibit a receiver from using other string as an alternative title to present the title information in a contents list.

- `sendHttpServerFileAs()` : Uploads a file on a storage device to an HTTP server.

Syntax

```
Array sendHttpServerFileAs(
    input String src_path,
    input String dest_path
)
```

Arguments

- `src_path` : URI representing the file to be copied
- `dest_path` : URI representing the directory or file on a http server, to which the specified file is uploaded.

Return values

- Array value that contains result of the operation
- 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
- 602: Invalid file name was specified
- 603: Not enough space available
- 604: Error during storing file
- 700: Service was disconnected
- NaN: Failure by other causes

Array[1]: Status-Code in HTTP1.1
 Array[2]: Byte length of received file
 Array[3]: Content-Type in HTTP1.1 (Media type described in header)

Description

The file on an http server specified in `src_path` is uploaded to the directory (or the file) on a http server, as specified in `dest_path`.

A return value is an array value consisting of Status-Code in HTTP1.1, the byte length of the sent file, and the media type. As the byte length of the sent file, the value of Content-Length used by http is returned. When the file is not successfully stored, Array [2] contains 0.

- `saveFtpServerFileAs()` : Copies a file on an FTP server to the specified path in a storage device.

Syntax

```
Number saveFtpServerFileAs(
    input String src_path,
    input String dest_path
)
```

Arguments

`src_path` URI representing the file on an FTP server to be copied
`dest_path` URI representing the file , to which the file specified in `src_path` is copied.

Return values

Array value that contains information on success or failure of the operation

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
-602:	Invalid file name
-603:	Not enough storage space available on the storage device
-604:	Error during storing file
-700:	Service was disconnected
NaN:	Failure by other causes

Array[1]:	Response Code in FTP
Array[2]:	Byte length of the received file

Description

The file in an FTP server specified in `src_path` is copied to the file (or directory) specified in `dest_path`.

A return value is an array value consisting of an error code, Response Code in FTP, and the byte length of the received file. When the file is not successfully stored, Array [2] contains 0.

- `saveFtpServerFile()` : Copies a file on an FTP server to an area in a storage device, as specified in a receiver's configuration.

Syntax

```
Number saveFtpServerFile(
    input String src_path
    [,input String content_title
    [,input String content_type
    [,input String drive_type
    [, input Number drive_number]]]]
)
```

Arguments

<code>src_path</code>	URI representing the resource to be copied
<code>content_title</code>	Content title used for listing contents
<code>content_type</code>	Media type of the content
<code>drive_type</code>	Type of the drive
<code>drive_number</code>	Integer representing the drive number (0 or larger)

Return values

Array value that contains the result of the operation

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

- 602: Invalid file name was specified
- 603: Not enough storage space available on the storage device
- 604: Error during storing file
- 700: Service was disconnected
- NaN: Failure by other causes

Array[1]: Response Code in FTP
Array[2]: Byte length of the received file

Description

The file specified in `src_path` is copied to an area, whose type is specified in `content_type` (When `content_type` is omitted, the destination area is the directory specified in a receiver's configuration. When `content_type` is available, the destination area is the directory whose type is specified a receiver's configuration.), of a drive, whose type is specified in `drive_type` (when `drive_type` is omitted, the type is specified based on a receiver's configuration). When the specified location is the existing directory, the receiver assigns a unique name that does not conflict with any existing directory name to a newly created area to store the copied information.

The available values of `drive_type` are:

- "InternalHDD" : Internal hard disk drive
- "MemoryCard" : Memory card
- "ExternalDevice" : External device

When `drive_number` is omitted, the default drive number specified in a receiver's configuration is used.

A return value is an array value consisting of an error code, Response Code in FTP, and the byte length of the received file. When the file is not successfully stored, Array [2] contains 0.

If `content_title` is omitted, no title information is retained by the destination file. Note that this does not prohibit a receiver from using other string as an alternative title to present the title information in a contents list.

- `sendFtpServerFileAs()` : Sends a file to an FTP server.

Syntax

```
Array sendFtpServerFileAs(
    input String src_path,
    input String dest_path
)
```

Arguments

`src_path` URI representing the file to be sent
`dest_path` URI representing the directory or file on a FTP server, to which the specified file is sent.

Return values

Array value that contains information on success or failure of the operation

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
- 602: Invalid file name was specified
- 603: Not enough storage space available on the storage device
- 604: Error during storing file
- 700: Service was disconnected
- NaN: Failure by other causes

Array[1]: Response Code in FTP

Array[2]: Byte length of the received file

Description

The file specified in src_path is sent to the directory (or the file) on a FTP server, as specified in dest_path.

- sendTextMail() : Sends a text mail.

Syntax

```
Array sendTextMail(
    input String subject,
    input String body,
    input String toAddress
    [, input String ccAddress]+
)
```

Arguments

subject	Subject of the mail
body	Body of the mail
toAddress	Address to which the mail is sent
ccAddress	Address to which copies of the mail is sent

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
- 700: Service was disconnected

NaN: Failure by other causes

Array[1]: Response Code in SMTP

Description

This function established a connection using SMTP to send an e-mail consisting of a subject and a body to an address specified in toAddress and ccAddress. The MIME-type of the e-mail body is text/plain. Whether or not the mail is successfully sent is indicated with a return value. Further information required to send the mail such as a parameter specifying a mail server is obtained from a receiver's configuration.

- 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: <table> <tbody> <tr> <td>"EUC-JP"</td> <td>EUC-JP</td> </tr> <tr> <td>"Shift_JIS"</td> <td>Shift-JIS</td> </tr> <tr> <td>"UTF-16"</td> <td>UCS/UTF-16</td> </tr> </tbody> </table>	"EUC-JP"	EUC-JP	"Shift_JIS"	Shift-JIS	"UTF-16"	UCS/UTF-16
"EUC-JP"	EUC-JP						
"Shift_JIS"	Shift-JIS						
"UTF-16"	UCS/UTF-16						

Return values

Array[0]:	Numeric value representing the result code <table> <tbody> <tr> <td>1:</td> <td>Success</td> </tr> <tr> <td>-1:</td> <td>Parameter error</td> </tr> <tr> <td>-2:</td> <td>Line was disconnected during transfer.</td> </tr> <tr> <td>-3:</td> <td>Time-out occurred.</td> </tr> <tr> <td>-300:</td> <td>Failed to establish an automatic connection</td> </tr> <tr> <td>-400:</td> <td>Failed to map names using DNS</td> </tr> <tr> <td>-500:</td> <td>Failed to process TLS-based operation</td> </tr> <tr> <td>NaN:</td> <td>Failure by other causes</td> </tr> </tbody> </table>	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
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[1]:	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.

- setDelayedTransmissionData(): Registers a data transmission to be performed at a specified time.

Syntax

```
Number setDelayedTransmissionData(
    input String regname,
    input String uri,
    input Date start_time,
    input Date end_time,
    input Number retry_interval,
    input String strtext,
    input String charset
)
```

Arguments

regname	Registration name of the delayed call						
uri	URI representing the location to which the data is sent						
start_time	Time to start the data transmission						
end_time	Time to end the data transmission						
retry_interval	Trial interval (in seconds)						
strtext	Text data to be sent						
charset	Character encoding used to send and receive the text data. The available values are:						
	<table> <tr> <td>"EUC-JP"</td><td>EUC-JP</td></tr> <tr> <td>"Shift_JIS"</td><td>Shift-JIS</td></tr> <tr> <td>"UTF-16"</td><td>UCS/UTF-16</td></tr> </table>	"EUC-JP"	EUC-JP	"Shift_JIS"	Shift-JIS	"UTF-16"	UCS/UTF-16
"EUC-JP"	EUC-JP						
"Shift_JIS"	Shift-JIS						
"UTF-16"	UCS/UTF-16						

Return values

1:	Success
-1:	Parameter error
-100:	Failure by specifying a past time
-101:	Failure by exceeding number of registration
-102:	Failure in writing the data into a restricted area
NaN:	Failure by other causes

Description

This function registers a data transmission to a server on the Internet, as specified in uri, to be performed at a time specified in start_time using TCP/IP.

If a data transmission (including a connecting process) fails, another transmission is performed when the period specified in retry_interval expires. A following transmission is performed when at least a period specified in retry_interval expires until the message is successfully sent and received. If the data has not been successfully sent and received until the time specified in end_time, the registered data transmission information is deleted and the failure is recorded as the result. In this case, no retrial occurs.

Unless a transmission/reception has failed, dialling is performed only once.

Note that if the data is in transmission at the time specified in `end_time`, the data transmission must be finished even it has passed the end time. The available values of an interval from `start_time` to `end_time`, the acceptable size of the text data, and the character encoding (charset) for the data transmission are defined in an operational standard regulation.

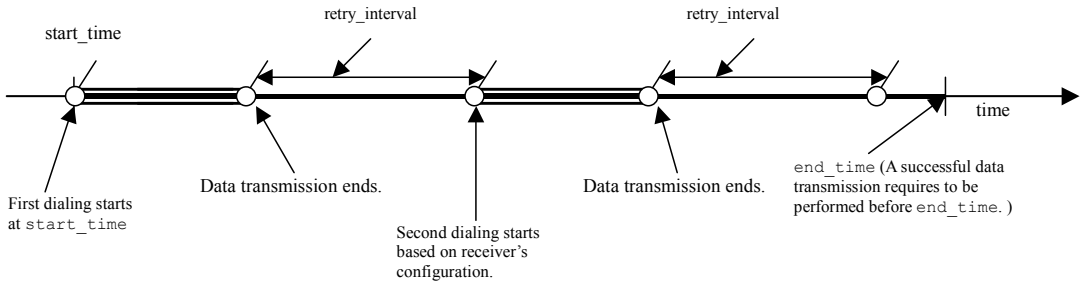


Figure 7-3 How `setDelayedTransmissionData()` works

The regname argument is used to present the transmitted data to human operators. Program identifications including `service_id` and `event_id` should be registered in order to identify the registering program. It is recommended that receiver be capable to display and delete information registered with the `setDelayedTransmissionData()` function.

- `sendMIMEEmail ()` : Sends multimedia data by e-mail.

Syntax

```
Array sendMIMEEmail(  
    input String subject,  
    input String src_module,  
    input String toAddress  
    [,input String ccAdress]+  
)
```

Arguments

subject	Subject of the multimedia data to be sent
src_module	Module containing MIME-encoded data as an entity
toAddress	Address to which the multimedia data is sent
ccAddress	Address that is not for an intended recipient of the multimedia data, but to which a "carbon copy" of the multimedia data is sent

Return values

Array[0]:	Values representing result codes
1:	Success
-1:	Parameter error
-2:	Line was disconnected during transmission
-3:	Time-out occurred
-300:	Failed to establish an automatic connection
-400:	Failed to map names using DNS

-700: Service was disconnected
NaN: Failure by other causes

Array[1]: Values representing SMTP response codes

Description

This function establishes a session using the SMTP protocol to send an e-mail consisting of subject and src_module to an address specified with toAddress. The src_module must contain a value representing a single module to which MIME-encoded data is mapped. Whether the data has been successfully sent or not is informed about through a return value. Other parameters including a mail server address and a sender's address are implicitly specified with information pre-configured in a receiver.

- setCacheResourceOverIP () : Cache resources on the Internet in the receiver.

Syntax

Number setCacheResourceOverIP(input Array resources)

Arguments

resources	An array containing URIs that identify resources on the Internet. Note that each of resources[0], resources [1], and resources[n] is of the String type to represent a URI that identifies a resource on the Internet.
-----------	--

Return values

1:	Success
NaN:	Failure

Description

This function stores information as specified in the argument resources, on resources on the Internet that can be kept in a cache in a receiver.

7.6.7.7 Status look-up functions for delayed call functions applicable to BASIC procedures and IP connections

- getDelayedTransmissionStatus():Obtains registered information about a data transmission to be performed at a specified time.

Syntax

Array getDelayedTransmissionStatus()

Arguments

None

Return values

String containing registered values:	Success
null:	Failure

Description

This function obtains the information about a data transmission to be performed at a specified time, which has been registered using a time-specified call function. The number of registered time-specified calls is represented as the number of elements of the returned Array object. When the number of registered time-specified calls is 0, an array of length 0 is returned. When the

Array object has one or more elements, each element, in turn, is an array object whose values are shown as below.

- Array[0] : Registration name (String)
- Array[1] : URI representing the location to which the data is sent. (String)
Note that an empty string is contained for a time-specified call that has been registered using setDelayedTransmissionDataOverBASIC().
- Array[2] : Time to start transmission (Date)
- Array[3] : Time to end transmission (Date)

- getDelayedTransmissionResult():Obtains a result of operating a registered time-specified call.

Syntax

Array getDelayedTransmissionResult()

Argument

None

Return values

String containing obtained values: Success

null: Failure

Description

This function obtains the result of operating a data transmission using a time-specified call function. The number of operated time-specified calls is represented as the number of elements of the returned Array object. When the number of operated time-specified calls is 0, an array of length 0 is returned. When the Array object has one or more elements, each element, in turn, is an array object whose values are shown as below.

- Array[0] : Registration name (String)
- Array[1] : URI representing the location to which the data is sent. (String)
Note that an empty string is contained for a time-specified call that has been registered using setDelayedTransmissionDataOverBASIC().
- Array[2] : Time at which the transmission was tried (Date)
- Array[3] : Numeric value representing the result code (Number)

7.6.7.8 Function for obtaining line connection status

- getPrefixNumber():Obtains information that has been registered by an end user for dialling.

Syntax

Array getPrefixNumber()

Argument

None

Return values

An Array object obtaining the information registered by a user for dialling is returned.

- Array[0]: The prefix numbers configured by the receiver
- Array[1]: The configured prefix numbers for dialling out (e.g. "0")

Array[2]:	The configured prefix numbers to disable or enable the notification of the caller number (e.g. "184"/"186")
Array[3]:	The configured prefix numbers to override the preferred carrier service assigned to the line ("122")
Array[4]:	The configured carrier identification ("00XY")

Description

This function obtains the configuration for dialling, which is retained in the receiver. The returned value is an Array object containing the configured values. When no value is configured or each set of the registered numbers are not recognized as intended, each of Arrays[0]-[4] contains an empty string.

7.6.7.9 Functions for operating root certificates for encrypted transmission

To establish encrypted transmission supported by TLS or SSL a root certificate that authenticates a broadcaster that operates Web servers is required. This section defines two functions for operating root certificates, which assume that any root certificate is transmitted through a data carousel. Root certificates are classified into the two types: generic root certificates and broadcaster-specific root certificates. A generic root certificate is stored persistently in a generic root certificate storage area in a receiver while a broadcaster-specific root certificate is valid only to a specific service and operated by the broadcaster. The term "root certificate" is a generic name that covers the two types. To express an operation specific to either type of the two, the terms "generic root certificate" and "broadcaster-specific root certificate" are used as required.

- isRootCertificateExisting() : Verifies whether a root certificate exists or not

Syntax

```
Number isRootCertificateExisting(
    input Number root_certificate_type,
    input Number root_certificate_id
    [,input Number root_certificate_version]
)
```

Arguments

root_certificate_type	Type of a root certificate (0: generic root certificate 1: broadcaster-specific root certificate)
root_certificate_id	root certificate identification number
root_certificate_version	root certificate version number

Return values

1 :	Success (the specified root certificate exists)
NaN :	Failure (the specified root certificate does not exist)

Description

This functions verifies whether the specified root certificate exists or not. The root_certificate_id and root_certificate_version arguments must be a signed 32-bit value. Details about root_certificate_id and root_certificate_version are defined in an operational standard regulation.

- getRootCertificateInfo() :Obtains information on a generic root certificate.

Syntax

Array getRootCertificateInfo()

Arguments

None

Return values

Array[0] : Contains information on a root certificate stored in Storage Area 0

Array[0][0] : root_certificate_id of a root certificate stored in Storage Area 0

Array[0][1] : root_certificate_version of a root certificate stored in Storage Area 0

Array[1] : Contains information on a root certificate stored in Storage Area 1

Array[1][0] : root_certificate_id of a root certificate stored in Storage Area 1

Array[1][1] : root_certificate_version of a root certificate stored in Storage Area 1

Apply the similar format to Array[2] through Array[6].

Array[7] : Contains information on a root certificate stored in Storage Area 7

Array[7][0] : root_certificate_id of a root certificate stored in Storage Area 7

Array[7][1] : root_certificate_version of a root certificate stored in Storage Area 7

null : Failure

Description

This functions obtains information on a generic root certificate stored in a generic root certificate storage area. A return value representing root_certificate_id or root_certificate_version must be a signed 32-bit value. A return value "0" indicates that the specified generic root certificate storage area has no generic root certificate.

7.6.8 Operational control functions

- reloadActiveDocument(): Reloads a BML 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 Section 9.2.6.

It is not ensured that the function verifies whether or not a stored program is currently played.

- lockExecution(): Instructs the current presentation to continue.

Syntax

Number lockExecution()

Argument

None

Return values

1: Success
NaN: Failure

Description

This function specifies that the presentation of a currently displayed BML document is kept presented after the end of the event. By default, the presentation of a BML document is aborted at the end of the event. Even if the continued presentation is explicitly specified, an engine event at the end of the broadcast program still occurs.

- unlockExecution(): Cancels a specified continued presentation.

Syntax

Number unlockExecution()

Argument

None

Return values

1:	Success
NaN:	Failure

Description

This function cancels a specified continued presentation of a BML document. If the event has already been finished at the time of this cancellation, the function immediately exits and moves to the next program.

- lockModuleOnMemory(): Receives a module into cache memory and locks the module.

Syntax

Number lockModuleOnMemory(input String module)

Argument

module	Module name
--------	-------------

Return values

1 :	Success
-1 :	Specified module does not exist.
-2 :	Cannot receive because of insufficient cash.
	(When a return value is 1, -1, or -2, the state can be confirmed using DII.)
NaN :	Failure by other causes

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 Chapter 9.

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 Chapter 9.

- 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 Chapter 9.

- getTuningLinkageSource(): Obtains a character string indicating the link source when the link descriptor was used to select the service.

Syntax

String getTuningLinkageSource()

Arguments

None

Return values

URI identifying the service:	Service was selected based on the link descriptor.
Empty string:	Service was selected independent of the link descriptor.

Description

This function returns an URI character string identifying a selected service if the function was invoked via a BML document that is part of the service and the service was selected with the link descriptor specified in ARIB STD-B10. The URI is described based on the conventions defined in Section 9.1.6. An empty string is returned when the service was selected independent of the link descriptor.

- `getTuningLinkageType()`: Obtains a linkage type when the link descriptor was used to select the service.

Syntax

Number `getTuningLinkageType()`

Argument

None

Return values

Numeric value identifying the linkage type:	Service was selected based on the link descriptor.
-1:	Service was selected independent of the link descriptor.

Description

This function returns a numeric value identifying the linkage type if the function was invoked via a BML document that is part of the service and the service was selected with the link descriptor specified in ARIB STD-B10. The description of linkage types complies with the conventions defined in ARIB STD-B10 Part 2, Chapter 2. The “-1” value is returned when the service was selected independent of the link descriptor.

- `getLinkSourceServiceStr()`: Obtains a character string indicating the service that is the source of a hyperlink.

Syntax

String `getLinkSourceServiceStr()`

Argument

None

Return values

URI character string indicating the service that is the source of a hyperlink :	Success
null :	Failure

Description

This function returns an URI character string indicating the service that is the source of a hyperlink. If the application has launched independent of hyper link descriptors, the function returns an empty character string. The description of returned URI conforms to the conventions on the namespace defined in Chapter 9.

- `getLinkSourceEventStr()`: Obtains a character string indicating the event that is source of a hyperlink.

Syntax

String `getLinkSourceEventStr()`

Argument

None

Return values

URI character string indicating the event that is the source of a hyperlink : Success
null : Failure

Description

This function returns an URI character string indicating the event that is the source of a hyperlink. If the application has launched independent of hyper link descriptors, the function returns an empty character string. The description of returned URI conforms to the conventions on the namespace defined in Chapter 9.

- 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 BML browser.

Syntax

Array getBrowserVersion()

Argument

None

Return values

Array[0]: String representing MakerID
Array[1]: String representing the name of BML 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 BML browser that controls presentation of the currently displayed BML 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 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 BML browser causes a minor version number to be updated. It is also recommended that when different types of receivers uses a same version of BML 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

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

null : Failure

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 listed in the following table.

Table 7-5 Applicable Values to type

type	Semantics
1	Event ID (event_id , a hexadecimal character string in the form of "0XXXXX")
2	Service ID(service_id , a hexadecimal character string in the form of "0XXXXX")
3	Network ID(original_network_id , a hexadecimal character string in the form of "0XXXXX")
4	Transport stream ID(transport_stream_id , a hexadecimal character string in the form of "0XXXXX")
5	Content ID(content_id, a hexadecimal character string in the form of "0XXXXXXXXXX")
6	Event reference (The notation conforms to the conventions on the namespace defined in Chapter 9.)
7	Service reference (The notation conforms to the conventions on the namespace defined in Chapter 9.)

type	Semantics
8	Content reference (The notation conforms to the conventions on the namespace defined in Chapter 9.)
9	Network ID(network_id, a hexadecimal character string in the form of “0XXXXX”)

When the type argument is specified as ‘5’ (Content ID) or ‘8’ (Content reference) for the content whose content_id is not specified, null (failure) is returned. If a value that is not listed in the table is specified, invoking an built-in function fails, and null is returned. The accuracy of specifying each identification depends on the conventions defined in the concerned broadcasting operational rules.

- getActiveDocument(): Returns the URI of a currently presented BML document.

Syntax

String getActiveDocument()

Argument

None

Return values

Character string that conforms to the conventions on the namespace defined in Chapter 9:

Success

null : Failure

Description

This function returns the URI of a currently presented BML 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 string used in sProvider and functionname is 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 BML document.

Syntax

```
Number launchDocument(  
    input String documentName,  
    input String transitionStyle  
)
```

Arguments

documentName	Character string to specify a BML document
transitionStyle	Transition style

Return values

1 :	Success
NaN :	Failure

Description

This function opens a BML 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 BML document under a restricted condition.

Syntax

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

Arguments

documentName	Character string to specify a BML document
transitionStyle	Transition style

Return values

1:	Success
NaN	Failure

Description

This function opens a BML 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 BML document over an interaction channel. Any BML document to which the documentName BML document transits based on this function or any further BML 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 BML document.

Syntax

```
Number quitDocument()
```

Argument

None

Return values

NaN

Description

This function quits presenting the specified BML document.

- launchExApp(): Launches an external application.

Syntax

```
Number launchExApp(
```

```

        input String uriname
        [, input String MIME_type
        [, input String Ex_info]+ ]
    )

```

Argument

uriname	URI
MIME_type	MIME type
Ex_info	String representing information required to allow an external application to process a specific media type of data.

Actual usage of Ex_info string is defined in an operational standard regulation.

Return values

1 :	Success (URI is successfully passed to an external application)
NaN :	Failure

Description

This function passes the data specified with URI to an external application that processes the data.

For the purpose of this standard, the term « external application » means a feature that processes a specific media type of data.

- 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, that is 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 shown below is supported by a receiver. Any string specified with mediaName is case sensitive. When an unknown string as mediaNam , 0 is returned.

The available values to mediaName of this function and linkMedia/Array[6] of linkMedia/Array[6] are shown below.

For future uses, all the strings not listed below are reserved.

String	Broadcasting Media Type
"1"	BS digital braodcast (11.7GHz ~ 12.2GHz)
"2"	Broadband CS digital broadcast (Right Circular Polarization)
"3"	Broadband CS digital broadcast (Left Circular Polarization)
"4"	Digital terrestrial broadcast
"5"	Digital terrestrial radio broadcast

- 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 is complies of the namespace convention defined in Section 9.2.11.

- 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 defined in Chapter 9.

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 exist, 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 Chapter 9.

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 a module that have been locked by lockModuleOnMemoryEx() is tried to unlock by unlockModuleOnMemory(), 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 module 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 is working on. Any such request for module is successfully cancelled.

- getLockedModuleInfo: Obtains a list of modules locked in a content memory.

Syntax

Array getLockedModuleInfo()

Arguments

None

Return values

Array containing information about modules:	Success
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

Apply the similar format to Array[2].

null: Failure

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 which 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.

- setFullDataDisplayArea() : Defines an area for presenting informative content transmitted through a data broadcasting service in a display in a receiver.

Syntax

Number setFullDataDisplayArea(input Number mode)

Argument

mode: Display configuration mode
 0 : Configures only part of the display as an area for presenting informative content
 1 : Configures the whole area of the display as an area for presenting informative content

Return values

1 : Success
 NaN : Failure

Description

This function is designed for services targeting portable terminals. This function configures the whole or a part of the usable display area in a receiver as an area for presenting text, graphics, and still pictures of a data broadcasting service (an area for presenting informative content). When either of the mode values has been specified, this function does not change the display configuration mode and returns NaN.

- getDataDisplayAreaSize() : Obtains the maximum number of characters in a vertical line or the maximum number of characters in a horizontal line in the current area for presenting informative content.

Syntax

Number getDataDisplayAreaSize(input String direction)

Argument

direction Specifies the direction of a line

Return values

Value representing the number of characters: Success
 NaN: Failure

Description

This function is used for services targeting portable terminals. The values applicable to direction are:

- H_size Requests the maximum number of characters in a horizontal line
- V_size Requests the maximum number of characters in a vertical line

When the direction argument contains H_size, the maximum number of characters in a horizontal line is returned. When the direction argument contains V_size, the maximum number of characters in a vertical line is returned.

- getBrowserStatus() : Obtains the status of a browser.

Syntax

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

Arguments

sProvider browser	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 additionalinfo. 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] : software	String representing a name of a resident application
Array[2] :	String representing a major version number
Array[3]:	String representing a minor version number
Array[4]: application	More information for an individual resident 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 (Refer to 4.1.1, ARIB STD-B24 Volume 2).

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 is three or less digits, leading 0s are required to be 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.

- startResidentApp(): Starts up a resident application software.

Syntax

```
Number startResidentApp(
    input String appName,
    input Number showAV,
    input String returnURI
    [, input String Ex_info])+
)
```

Argument

appName

Name of a resident application software to be started up

showAV

Flag that specifies whether or not the current playback of a TV program (video and sound) is allowed to continue when the resident application software has been started up

1: The playback is allowed to continue

0: The playback is not allowed to continue

returnURI

URI of a component that is rendered first when the BML browser is restarted after the resident application software that was started up by the function has been quitted. To specify no component, returnURI must contain an empty string. This argument is designed to help a receiver to work. It is not required that any receiver depends on the argument to work properly.

Ex_info

String representing more information on starting up a resident application software

Return values

1:	Success
NaN :	Failure

Description

This function starts up an resident application software, as specified in appName. Valid combinations of values applicable to the arguments appName, showAV, and Ex_info are defined in an operational standard regulation. Once this function is executed, no script parts following this function are executed, the data broadcasting engine quits, and the control is passed to the resident application software. When the resident application software quits, a process is required to select again the service that was interrupted by the execution of this function.

7.6.9 Receiver sound control

- 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 defined in Chapter 9.

7.6.10 Timer functions

- 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).

- setTimeout(): Performs a function or processing command when a time specified in milliseconds expires.

Syntax

```
Number setTimeout(
    input String func,
    input Number interval
)
```

Arguments

func	Command or function name executed by this function.
interval	Interrupt interval (in milliseconds)

Return values

Positive value :	Registration timer ID
NaN :	Failure

Description

This function invokes a function or command when a time specified with interval (in milliseconds) expires.

- 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 specified interval with msec for the number of times specified with iteration. If iteration is 0 (zero), the invocation is repeated until clearInterval is called.

- clearTimeout(): 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 sleep function, other functions are not affected.

This function applicable to a timer generated by setTimeout() or 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 setTimeout() or 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. 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. And it is assumed that when playing a stored-stream content, a clock that is based on PartialTS Time Descriptor of SIT is referenced. This function affects only to the document group(specified in the Chapter 4 of Appendix 1). At an initial state immediately after the load of a document, Date() operates based on the absolute playback time.

7.6.11 External character functions

loadDRCS() : config 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 conventions defined in Chapter 9.

The loaded external character data is effective until unloadDRCS() is called or the display of a BML document ends. The content referenced by DRCS_ref conforms to the format conventions described in ARIB STD-B24 Volume 1, Appendix D.

- unloadDRCS() : clear external character data

Syntax

Number unloadDRCS()

Arguments

None

Return values

1 :	Success
NaN :	Failure

Description

This function clears the external character configuration.

7.6.12 Functions for controlling external devices

- enumPeripherals(): Enumerates external devices.

Syntax

Array enumPeripherals([input String public_Identifier])

Argument

public_Identifier	Formal public identifier
-------------------	--------------------------

Return values

Array object contains the result :	Success
null :	Failure

Description

This function enumerates the external devices that are connected to be ready for exchanging data by using an XML document described with DTD that is a formal public identifier specified in the argument public_Identifier. Each element of the returned Array object is a set of string representing URI of an external device that was obtained by this function. The Description of URI complies with the namespace conventions defined in Section 9.2.17.

When public_Identifier is omitted, the available external devices are enumerated.

- passXMLDocToPeripheral(): Passes an XML document to an external device.

Syntax

```
Number passXMLDocToPeripheral(
    input String peripheral_Ref,
    input String XML_ref
    [,input String messageString]
)
```

Arguments

peripheral_Ref	URI of an external device
XML_ref	URI of an XML document
messageString	Message string to be passed to the external device

Return values

1 :	Success
NaN :	Failure

Description

This function passes an XML document file contained in a location identified with URI specified in the XML_ref argument to an external device whose URI is specified in the peripheral_Ref argument. The description of peripheral_Ref complies with the namespace conventions defined in Section 9.2.17. The description of XML_ref complies with the namespace conventions defined in Section 9.2.

- getArrayFromPeripheral () : Communicates with external devices.

Syntax

```
Array getArrayFromPeripheral (
    input String peripheral_Ref,
    input String method,
    input Number timeout
    [, input Array data]
)
```

Arguments

peripheral_Ref	URI of an external device
method	Format of an intended set of information
timeout	Specifies a period of time (in milliseconds). When the specified period expires, the process is assumed to be timed-out.
data	Specifies information to be input into an external data as an array

Return values

Array[0] :	Values representing results
1:	Success
-1:	Parameter error
-2:	Method contains an unsupported value
-3:	Time-out occurred
NaN :	Failure by other causes
Array[1] :	This array and following arrays contain values that has been read out

Description

This function passes information as an array, as specified with data, to an external device of which URI is specified with peripheral_Ref and returns the result in a series of Array objects. URI formats applicable to peripheral_Ref comply with Section 9.2.17.

7.6.13 Functions for controlling bookmark areas

- writeBookmarkArray(): Writes to a bookmark area.

Syntax

```
Number writeBookmarkArray(
    input String filename,
    input String title,
    input String dstURI,
    input String expire_str,
    input String bmType,
    input String linkMedia,
    input String usageFlag
    [,input String extendedStructure,
    input Array extendedData]
)
```

Arguments

filename	URI representing a bookmark area
title	Title of the bookmark
dstURI	URI representing a location to which the bookmark links
expire_str	Expiration Data of the bookmark
bmType	Type of the bookmark
linkMedia	Type identification of the link destination
usageFlag	Type of usage
extendedStructure	Type specification for each element of extended data
extendedData	Array of data to be contained in extended data

Return values

1 :	Success
NaN :	Failure

Description

This function writes information including a link destination specified with dstURI to a bookmark area specified with filename. title is a string outlining the bookmark. expire_str is a 10-digit string with the format "YYYYMMDD HH" indicating the expiration date of the bookmark. bmType is a string, whose specification is defined in an operational standard regulation, representing the type of the bookmark. linkMedia contains a string representing the media of the link destination. usageFlag is a flag indicating whether or not the access to the bookmark area requires data in the extended data area. The available values to usageFlag are:

- "0": The bookmark area is not accessible when the specified type of extended data area is not available.
- "1": The bookmark area is accessible even when the specified type of extended data area is not available.

A bookmark area consist of a basic data area and an extended data area. The structure of a basic data area is common to any bookmark area. When extendedStructure and extendedData are omitted, the extended data area is not written. The available structure of an extended data per bookmark type is defined in an operational standard regulation.

The format of an array to be contained in an extended data area, that is specified with extendedStructure and extendedData complies with the format of structure of BinaryTable defined in section 7.5.2.2, expect that the record length is not recorded. The structure and type are specified in the following formats, for the purpose of this function:

```
structure ::= field[" , " field]*
type ::= "B" | "U" | "I" | "S" | "P"
```

The information described with the above arguments is written. On RECEIVER a receiver side, the date and time when the writing access is executed and a flag indicating nondeletability are written. The written date and time are read out into Array[3] of the returned value of the readBookmarkArray() function. The nondeletability flag is obtained in Array[4]. When this flag contains "0", the written information is deletable. When this flag contains "1", the written information is nondeletable. This flag is set to "1" by the lockBookmark() function and set to "0" by the unlockBookmark() function.

The bookmark area specified with filename has been written, this function is allowed to overwrite it provided that the specified title, dstURI, and bmType are equivalent to the corresponding information that has been contained in the bookmark area.

- readBookmarkArray(): Reads data from a bookmark area.

Syntax

```
Array readBookmarkArray(
    input String filename
    [,input String bmType
    ,input String extendedStructure]
)
```

Arguments

filename	URI representing a bookmark area
bmType	Type of the bookmark
extendedStructure	Type specification for each element of extended data

Return values

Array object containing the values that was read out : Success

```
Array[0]: Title (String)
Array[1]: Link destination URI (String) representing a location to which the bookmark
links
Array[2]: Expiration date in "YYYYMMDDHH" format (String)
Array[3]: Registerd date and time in "YYYYMMDDHH" (String)
Array[4]: Deletability flag (String)
           "0":   deletable
           "1":   nondeletable
Array[5]: Type of bookmark (String)
```

- Array[6]: Link destination identification (String)
- Array[7]: Type of usage (String)
- Array[8]: Array of type specifications for elements of extended data
(Array) or null (Type specification is omitted, Different type is specified)

null: Failure (No information is written)

Description

This function read the bookmark data that has been written by the writeBookmarkArray() function from the bookmark area specified with filename to obtain the returned values as an Array object. The description of filename complies with the namespace conventions defined in Section 9.2.10. When no information is written, null is returned. Despite of the existence of a value of bmTyp, the information in the basic data area is obtained in the returned values Array[0]-[7]. When bmType is specified and is equivalent to the recorded bookmark type, the information written to the extended data area written by the writeBookmarkArray() function is read out based on an individual element type that is specified with extendedStructure, and returned as an extended data array in Array [8].

- deleteBookmark(): Deletes a bookmark.

Syntax

Number deleteBookmark(input String filename)

Argument

filename	URI representing a bookmark area
----------	----------------------------------

Return values

1 :	Success
NaN :	Failure

Description

This function deletes the information in a bookmark area specified with filename. When a bookmark area with no information written or the information is specified as nondeletable, NaN is returned.

- lockBookmark(): Specifies a bookmark area as a nondeletable area.

Syntax

Number lockBookmark(input String filename)

Argument

filename	URI representing a bookmark area
----------	----------------------------------

Return values

1 :	Success
NaN :	Failure

Description

This function defines a bookmark area specified with filename as a nondeletable area. When a bookmark area with no information written, NaN is returned.

Otherwise, the flag indicating nondeletability "1" is recorded.

Note that when this function locks a bookmark area that has been locked, Success is returned.

- unlockBookmark(): Specifies a bookmark area as a deletable area.

Description

It is recommended that a BML content keeps working as intended after the bookmark list application has been successfully started. How a presentation of a BML content is kept or interrupted depends on an implementation.

While a bookmark list application, which is a receiver's feature is presented to an end user, any input through keys on a remote control is obtained by the bookmark list application.

When a transition from a bookmark list to a destination of a bookmarked link fails and the bookmark list application exits, the previously presented a BML content appears again and any input through keys on a remote control is obtained by the BML content.

This function must be an asynchronous function.

7.6.14 Other functions

- 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 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 a time in 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 add
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 add time in unit specified by unit to base and returns the result. 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).

7.6.15 Ureg pseudo object properties

- Ureg: Browser Pseudo Object Properties

Syntax

Ureg[0], Ureg[1], ... Ureg[63]

Description

The Ureg property retains 64 values that are numbered from 0 to 63. These values are stored in a locked area that has been reserved by the system. The following restrictions are applied to these values.

Values are character string type only. The maximum size of character string is 256 bytes.

The Ureg property must be unique in the system.

The initial value stored in Ureg is an empty string. And, if the byte length of the specified string exceeds 256, up to 256 bytes are stored. If the 256th byte is the first byte of a 2-bytes character code, the character is not stored.

7.6.16 Greg pseudo object properties

- Greg: Browser Pseudo Object Properties

Syntax

Greg[0], Greg[1], , , , , Greg[63]

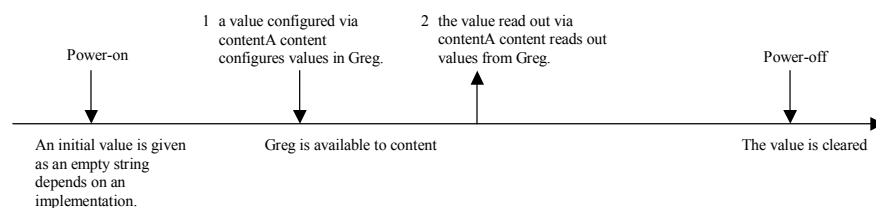
Description

The Greg property retains 64 values that are numbered from 0 to 63. These values are stored in a locked area that has been reserved by the system. The following restrictions are applied to these values.

Values are character string type only. The maximum size of character string is 256 bytes.

Unlike Ureg, Greg supports different media. Greg is an area that retains any value configured via a content from the period from a power-on to a power-off and that is allowed to be overwritten as needed

The Greg property must be unique in the system.



The initial value stored in Greg is an empty string. And, if the byte length of the specified string exceeds 256, up to 256 bytes are stored. If the 256th byte is the first byte of a 2-bytes character code, the character is not stored.

7.6.17 Functions for Printing

- getPrinterStatus() : Obtains a status of a printer.

Syntax

Number getPrinterStatus([input String MIME_type])

Arguments

MIME_type MIME type of a document to be printed

Return values

2 :	The printer is ready to print and can obtain resources on the Internet.
1 :	The printer is ready to print but cannot obtain resources on the Internet.
-1 :	Parameter error
-2 :	The printer has not responded due to disconnection or offline
-3 :	Error due to a full buffer or other auto-recoverable but time-consuming troubles
-4 :	Error due to a paper jam in the concerned printer or other troubles that require human operation to recover
-5 :	The MIME type of the document to be printed is not supported by the printer
NaN :	Other errors

Description

This function obtains the current status of the printer. When the function includes the MIME_type argument, the function also inquire whether or not the specified MIME type is supported.

- printFile() : Prints out a file consisting of static data.

Syntax

```
Number printFile(
    input String resource_ref
    [,input String module_ref]+
)
```

Arguments

resource_ref	URI of a document to be printed
module_ref	URI of a module that comprises a document to be printed

Return values

1 :	Success
-1 :	Parameter error
-2 :	The printer has not responded due to disconnection or offline
-3 :	Error due to a full buffer or other auto-recoverable but time-consuming troubles
-4 :	Error due to a paper jam in the printer or other troubles that require human operation to recover
-5 :	The MIME type of the document to be printed is not supported by the printer
-6 :	Failure in obtaining a specified module
NaN :	Other errors

Description

This function obtains a document to be printed via a broadcasting service and transfers the document to a printer to print out the document. The `resource_ref` argument contains a value specifying a document to be printed while the `module_ref` argument contains a value specifying a module containing resources (including a value in `resource_ref`) that construct the document to be printed.

- `printTemplate()` : Prints out a file consisting of dynamic data.

Syntax

```
Number printTemplate(
    input String resource_ref,
    input Array keyword
    [,input String module_ref])+
)
```

Arguments

<code>resource_ref</code>	URI of a template document to be printed
<code>keyword</code>	Associative array that has key strings to be replaced and value strings
<code>module_ref</code>	URI of a module that constructs a document to be printed

Return values

1 :	Success
-1 :	Parameter error
-2 :	The printer has not responded due to disconnection or offline
-3 :	Error due to a full buffer or other auto-recoverable but time-consuming troubles
-4 :	Error due to a paper jam in the printer or other troubles that require human operation to recover
-5 :	The MIME type of the document to be printed is not supported by the printer
-6 :	Failure in obtaining a specified module
NaN :	Other errors

Description

This function obtains a document to be printed via a broadcasting service, then edits the document dynamically, and transfers the document to a printer to print out the document. For a successful operation of the function, a template document is required. A template document contains keywords, each of which will be replaced with a value string by a script. This allows a document to be dynamically configured and printed out. When a template file contains a string `&keyword;` a property value which has the same name as `keyword` in an associative array specified with the `keyword` argument. The `resource_ref` argument contains a value specifying a document to be replaced while the `module_ref` argument contains a value specifying a module containing resources (including a value in `resource_ref`) that comprise the document to be printed.

- `printUri()` : Prints out a document that exists at a specified URL

Syntax

```
Number printUri(input String uri)
```

Argument

uri URI at which a document to be printed exists

Return values

1 : Success
-1 : Parameter error
-2 : The printer has not responded due to disconnection or offline
-3 : Error due to a full buffer or other auto-recoverable but time-consuming troubles
-4 : Error due to a paper jam in the printer or other troubles that require human operation to recover
-5 : The printer cannot obtain the resources on the Internet
NaN : Other errors

Description

This function allows a file that exists at a specified URI to be printed out. A receiver transfers the URI information specified with the argument to a printer. A printer is assumed to be responsible for obtaining the other resources needed for this printing task via a certain communication.

- printStaticScreen() : Prints out an image on a composition of still picture plane and text and graphics planes.

Syntax

Number printStaticScreen([input Number pattern])

Arguments

pattern Printing layout pattern

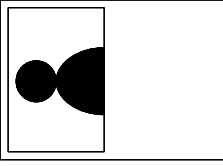
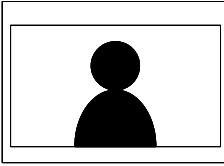
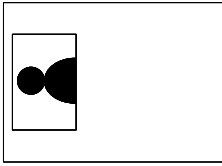
Return values

1 : Success
-1 : Parameter error
-2 : The printer has not responded due to disconnection or offline
-3 : Error due to a full buffer or other auto-recoverable but time-consuming troubles
-4 : Error due to a paper jam in the printer or other troubles that require human operation to recover
-5 : The MIME type of the concerned document to be printed is not supported by the printer
NaN : Other errors

Description

This function composes still picture planes and text and graphics planes and transfers the composition to a printer to print out the composition. The following table shows the values applicable to the pattern argument and their corresponding printing layout patterns. When the pattern argument contains no value, "2" is assumed to be specified.

Value for pattern	2	1	3
-------------------	---	---	---

Value for pattern	2	1	3
Printing layout pattern			
Ratio of printable area to image	1/2	1	1/4

- saveImageToMemoryCard() : Saves a image file into a memory card.

Syntax

```
Number saveImageToMemoryCard(
    input String src_resource,
    input String dst_filename,
    input Boolean overwrite
)
```

Arguments

src_resource	URI of an image file to be saved
dst_filename	File name of the saved image, which replaces a name specified with src_resource
overwrite	Specifies whether the concerned image file is overwritable or not (true : overwritable, false : non-overwritable)

Return values

1 :	Success
-1 :	No memory card
-2 :	Write protected
-3 :	No available space in the inserted memory card
-4 :	The overwrite argument contains "false" and a file specified with dst_filename has been saved
-5 :	Failure in obtaining an image file to be saved
NaN :	Other errors

Description

This function saves a file specified with src_resource into a memory card specified with dst_filename. The dst_filename must contain a file name consisting of eight characters followed by a dot and three characters. The src_resource argument must contain a value representing a resource in a data carousel.

- saveHttpServerImageToMemoryCard() : Saves a image file into a memory card.

Syntax

```
Number saveHttpServerImageToMemoryCard(
    input String src_resource,
    input String dst_filename,
```

input Boolean overwrite

)

Arguments

src_resource	URI of an image file to be saved
dst_filename	File name of the saved image, which replaces a name specified with src_resource
overwrite	Specifies whether the image file is overwritable or not (true : overwritable, false : non-overwritable)

Return values

1 :	Success
-1 :	No memory card
-2 :	Write protected
-3 :	No available space in the inserted memory card
-4 :	The overwrite argument contains "false" and a file specified with dst_filename has been saved
-5 :	Failure in obtaining an image file to be saved
-6 :	Line was disconnected during the file transfer
-7 :	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 :	Other errors

Description

This function saves a file specified with src_resource into a memory card specified with dst_filename. The dst_filename argument must contain a file name consisting of eight characters followed by a dot and three characters. The src_resource argument must contain a value representing a resource to be obtained via an IP channel.

- saveStaticScreenToMemoryCard() : Saves an image file on a composition of still picture planes and text and graphic planes into a memory card.

Syntax

```
Number saveStaticScreenToMemoryCard(  
    input String dst_filename,  
    input Boolean overwrite  
)
```

Arguments

dst_filename	File name of the saved image, which replaces a name specified with src_resource
overwrite	Specifies whether the image file is overwritable or not (true : overwritable, false : non-overwritable)

Return values

1 :	Success
-1 :	No memory card
-2 :	Write protected
-3 :	No available space in the inserted memory card
-4 :	The overwrite argument contains "false" and a file specified with dst_filename has been saved
NaN :	Other errors

Description

This function saves an image on a composition of currently presented still picture planes and text and graphics planes into a memory card. The saved image file has a name specified with dst_filename. The dst_filename argument must contain a file name consisting of eight characters followed by a dot and three characters.

7.6.18 Server-based broadcasting functions

7.6.18.1 Storage schedule functions

- epgStoreReserve(): Schedules a storing process of a program.

Syntax

```
Number epgStoreReserve(
    input String event_ref
    [,input Date start_time]
)
```

Arguments

event_ref	Specifies an event
start_time	Start time of a specified event

Return values

1:	Success
NaN:	Failure

Description

This function schedules a storing process of an event, as specified in the argument event_ref, which is scheduled to start at the time designated by start_time. Success or failure is returned by the returned value.

The description of event_ref conforms to the namespace conventions defined in Chapter 9. If start_time is omitted, this function acts on the event specified in event_ref.

- epgStoreCancelReservation(): Cancels the scheduled storing process a specified event.

Syntax

```
Number epgStoreCancelReservation(input String event_ref)
```

Arguments

event_ref	Specifies an event
-----------	--------------------

Return values

1:	Success
----	---------

NaN: Failure

Description

This function cancels the scheduled storing process of an event designated by `event_ref`. Success or failure is returned by the return value. The description of `event_ref` conforms to the namespace conventions defined in Chapter 9.

- `epgStoreCheckReservation()`: Verifies how an event is scheduled to be stored.

Syntax

```
Number epgStoreCheckReservation(
    input String event_ref
    [,input Date start_time]
)
```

Arguments

<code>event_ref</code>	Specifies an event
<code>start_time</code>	Start time of the specified event

Return values

2: The specified event has been scheduled to be stored


1: The specified event has not been scheduled to be stored (due to conflicting storing schedules)

0: The specified event has not been scheduled to be stored (with no conflicting storing schedules)

NaN: Failure

Description

This function verifies whether or not an event designated by `event_ref` has been scheduled to be stored, and whether or not there are conflicting schedules. The result is returned by the return value. The description of `event_ref` conforms to the namespace conventions defined in Chapter 9.

- `seriesStoreReserve()`: Schedules a storing process of a series of rams.

Syntax

```
Number seriesStoreReserve(
    input String series_ref,
    input Date expire_date
)
```

Arguments

<code>series_ref</code>	Specifies a series of programs
<code>expire_date</code>	Expiration date of the specified series of programs

Return values

1: Success

NaN: Failure

Description

This function schedules a storing process of a series of programs, as specified in the argument `series_ref`. Success or failure is returned by the returned value. Note that if the absolute time at which this function is executed is after what `expire_date` contains, NaN is returned to imply that the value in `series_ref` is invalid. The description of `series_ref` conforms to the namespace conventions defined in Chapter 9.

- `seriesStoreCancelReservation()`: Cancels the scheduled storing process of a specified series of programs.

Syntax

```
Number seriesStoreCancelReservation(  
    input String series_ref,  
    input Date expire_date  
)
```

Arguments

<code>series_ref</code>	Specifies a series of programs
<code>expire_date</code>	Expiration date of the specified series of programs

Return values

1:	Success
NaN:	Failure

Description

This function cancels the scheduled storing process of a series of programs designated by `series_ref`. Success or failure is returned by the return value. Note that if the absolute time at which this function is executed is after what `expire_date` contains, NaN is returned to imply that the value in `series_ref` is invalid. The description of `event_ref` conforms to the namespace conventions defined in Chapter 9.

- `seriesStoreCheckReservation()`: Verifies how a series of programs is scheduled to be stored.

Syntax

```
Number seriesStoreCheckReservation(  
    input String series_ref,  
    input Date expire_date  
)
```

Arguments

<code>series_ref</code>	Specifies a series of programs
<code>expire_date</code>	Expiration date of the specified series of programs

Return values

2:	The specified series of programs has been scheduled to be stored
1:	The specified series of programs has not been scheduled to be stored (due to conflicting storing schedules)
0:	The specified event has not been scheduled to be stored (with no conflicting storing schedules)

NaN: Failure

Description

This function verifies whether or not a series of programs designated by `series_ref` has been scheduled to be stored, and whether or not there are conflicting schedules. The result is returned by the return value. Note that if the absolute time at which this function is executed is after what `expire_date` contains, NaN is returned to imply that the value in `series_ref` is invalid. The description of `event_ref` conforms to the namespace conventions defined in Chapter 9.

7.6.18.2 Storage functions

- `storeStart()`: Starts to a storing process of a carousel.

Syntax

Number `storeStart(input String es_ref)`

Arguments

`es_ref` **Specifies an ES**

Return values

- 1: Success in starting the storing process
- 800: Failure because the specified ES has not been found (Failure because there is no available carousels although the specified ES exists)
- 801: Failure because the currently transmitted carousel is empty
- 810: Failure because no receiving functions on the receiver are available
- 811: Failure because there is not enough storage capacity on the receiver
- NaN: Failure due to other causes

Description

This function commands to start a storing process of an ES, as specified in `es_ref`. Success or failure is returned by the return value. The description of `es_ref` conforms to the namespace conventions defined in Chapter 9.

- `storeTerminate()`: Terminates a storing process of a carousel.

Syntax

Number `storeTerminate(input String es_ref)`

Arguments

`es_ref` **Specifies an ES**

Return values

- 1: Success
- NaN: Failure

Description

This function terminates a storing process of an ES, as specified in `es_ref`. Success or failure is returned by the return value. The description of `es_ref` conforms to the namespace conventions defined in Chapter 9.

- **checkStoreStatus()**: Verifies whether or not a storing process of a carousel is active.

Syntax

Number checkStoreStatus(input String es_ref)

Arguments

es_ref Specifies an ES

Return values

1: The storing process is active(The storing process has been commanded to start)
0: The storing process is inactive
NaN: Failure

Description

This function returns a value to inform whether or not the storing process of the ES designated by es_ref is active. The description of es_ref conforms to the namespace conventions defined in Chapter 9.

7.6.18.3 License functions

- **getLicense()**: Obtains or updates a license.

Syntax

```
Array getLicense(
    input String content_path,
    input String provider_id,
    input String license_id,
    input String src_path,
    input Number type
)
```

Arguments

content_path URI of server-based content
provider_id Identification of a broadcaster or service provider
license_id License identification
src_path URI of a communication server in which the license resides
type Process type
0:Process for obtaining the license
1:Process for updating the license

Return values

Array containing the result: Success
Array[0]:Indicates whether or not a CAS client properly works
0: Normal termination
-860: No CAS clients are available
-861: Found CAS clients are invalid
-862: Error on a CAS client
-880: No license to be updated

NaN: Failure due to other causes

Array[1]:Result of what has been executed via a CAS client

null: Failure

Description

This function obtains a license designated by provider_id or license_id from a communication server designated by src_path, or updates a license designated by provider_id or license_id that resides in a communication server designated by src_path. The result of the obtaining process or the updating process is returned as the return value. The function uses Array[1] to show the result of what has been executed via a CAS client. Details on how the result is treated are defined in an operational standard regulation.

- getLicenseIDList(): Obtains a list of license IDs.

Syntax

```
Array getLicenseIDList(  
    input Number page,  
    input Number search_type,  
    input String provider_id,  
    input String license_id  
)
```

Arguments

pageNumber representing a page to be obtained

search_type Search type

provider_id Broadcaster/Service provider ID used as a filtering condition

license_id License ID used as a filtering condition

Return values

Array containing information: Success

Array[0]:Indicates whether or not a CAS client properly works

0:Normal termination

-860:No CAS clients are available

-861:Found CAS clients are invalid

-862>Error on a CAS client

NaN:Failure due to other causes

Array[1]:Result of what has been executed via a CAS client

Array[2] and following elements:License IDs and more information

null: Failure

Description

This function obtains a list containing license IDs and information on statuses of licenses, as specified in search_type, provider_id, and license_id, in a page format, as specified in page. Values applicable to search_type and how the result of what is executed via a CAS client is treated are defined in an operational standard regulation. Details on Array[2] and

following elements, and their values are also defined in an operational standard regulation.

- getLicenseStatus(): Obtains usage conditions and usage statuses of licenses.

Syntax

```
Array getLicenseStatus(
    input String provider_id,
    input String license_id,
    input String license_handle,
    input Number search_type
)
```

Arguments

provider_id	Identification of a broadcaster or service provider
license_id	License identification
license_handle	Identification of an instance of a license
search_type	Search type

Return values

Array containing information: Success

Array[0]: Indicates whether or not a CAS client properly works

0: Normal termination

-860: No CAS clients are available

-861: Found CAS clients are invalid

-862: Error on a CAS client

-881: No license for storing into a receiver, as specified

NaN: Failure due to other causes

Array[1]: Result of what has been executed via a CAS client

Array[2] and following elements: Information on usage conditions and usage statuses of licenses

null: Failure

Description

This function obtains arrays containing information on usage conditions and usage statuses of licenses, as specified in search_type, license_id, and license_handle. Values applicable to search_type and how the result of what is executed via a CAS client is treated are defined in an operational standard regulation. Details on Array[2] and following elements, and their values are also defined in an operational standard regulation.

- setPurchaseInfo(): Registers information on agreements (purchases) of licenses in the receiver.

Syntax

```
Number setPurchaseInfo(
    input String id,
    input Date valid_date,
```

```
    input Number type,  
    input Array license_list  
)
```

Arguments

id Identification of a group of server-based content components
valid_date Effective (Expiration) time and date of a license
type Agreement (Purchase) type
license_list List of license IDs

Return values

1: Success
NaN: Failure

Description

This function registers a license ID representing a group of server-based content components and information on agreements (or purchases) of licenses into the receiver. The function accepts an identification of a group that consists of two or more server-based content components as the argument id. Details on the argument id are defined in an operational standard regulation. The information entered by this function is stored in a non-volatile memory in the receiver. The argument license_list is designed to contain license IDs relating to the specified group. Details on the argument license_list are defined in an operational standard regulation. The argument type assumes values representing types of agreement statuses and the argument valid_date is provided to contain dates relating to such agreements. Details on the arguments type and valid_date are also defined in an operational standard regulation.

- getPurchaseInfo(): Obtains information on agreements (purchases) of licenses.

Syntax

Array getPurchaseInfo(input String id)

Arguments

id Identification of a group of server-based content components

Return values

Array containing information: Success
Array[0] and following elements: License IDs
null: Failure

Description

This function obtains information on agreed (purchased) license IDs, which has been entered into the receiver by the function setPurchaseInfo(). The function accepts an identification of a group that consists of two or more server-based content components as the argument id. Details on the argument id are defined in an operational standard regulation. When no license IDs applicable to a group, as specified in the argument id, has been found, an Array object of length 0 is returned.

- getLicenseLinkInfo(): Obtains information on licenses

Syntax

Array getLicenseLinkInfo(input String content_path)

Arguments

content_path URI of server-based content

Return values

Array containing information: Success

null: Failure

Description

This function obtains information on licenses for server-based content, as specified in content_path. Details on arrangements and values of returned arrays containing information are defined in an operational standard regulation.

- getEncryptionkeyInfo(): Obtains information on content keys.

Syntax

Array getEncryptionkeyInfo(input String content_path)

Arguments

content_path URI of server-based content that has been stored

Return values

Array containing information: Success

null: Failure

Description

This function obtains information on content keys for the server-based content, as specified in content_path. Details on arrangements and values of returned arrays containing information are defined in an operational standard regulation.

7.6.18.4 CAS functions

- requestCASProcess(): Requests a CAS client to execute a process.

Syntax

```
Array    requestCASProcess(  
         input Number process_type,  
         input Array input_data  
         )
```

Arguments

process_type Process type

input_data Arguments to be passed to a CAS client

Return values

Array containing the result: Success

Array[0]: Indicates whether or not a CAS client properly works

0: Normal termination

-860: No CAS clients are available

-861:Found CAS clients are invalid

-862:Error on a CAS client

NaN:Failure due to other causes

Array[1]:Result of what has been executed via a CAS client

null: Failure

Description

This function requests a CAS client to execute a process, as specified in process_type. Details on results of what has been executed via a CAS client, and values applicable to the argument process_type and input_data are defined in an operational standard regulation.

- getCASInfo(): Requests a CAS client to offer information and obtains the information in a returned value.

Syntax

```
Array getCASInfo(  
    input Number data_type,  
    input Array input_data  
)
```

Arguments

data_type Type of information to be obtained

input_data Data to be passed to a CAS client

Return values

Array containing information: Success

Array[0]:Indicates whether or not a CAS client properly works

0:Normal termination

-860:No CAS clients are available

-861:Found CAS clients are invalid

-862:Error on a CAS client

-880:No license to be updated

NaN:Failure due to other causes

Array[1]:Result of what has been executed via a CAS client

Array[2]:Information obtained via a CAS client

null: Failure

Description

This function requests a CAS client to offer information of a type designated by data_type. Details on values applicable to data_type and input_data are defined in an operational standard regulation.

7.6.18.5 Server-based content control functions

- isContentStored(): Verifies whether or not specified server-based content has been stored.

Syntax

Boolean isContentStored(input String content_path)

Arguments

content_path URI of server-based content to be verified

Return values

true: The specified server-based content has been stored

false: The specified server-based content has not been stored

Description

This function verifies whether or not the server-based content designated by content_path has been stored.

- lockStoredContent(): Locks a server-based content.

Syntax

Number lockStoredContent(input String content_path)

Arguments

content_path URI of server-based content to be locked

Return values

1: Success

-1: Parameter error

-820: The specified server-based content has not been found

-821: No more server-based content can be locked

-822: Unable to lock due to other processes (including a process that is storing the same server-based content)

-823: The specified server-based content has been locked

NaN: Failure due to other causes

Description

This functions locks server-based content designated by content_path to prevent the resource from being deleted.

- unlockStoredContent(): Unlocks locked server-based content.

Syntax

Number unlockStoredContent(input String content_path)

Arguments

content_path URI of server-based content to be unlocked

Return values

1: Success

-824: The specified server-based content has not been locked

NaN: Failure due to other causes

Description

This function unlocks server-based content that has been locked.

- isLockedStoredContent(): Verifies whether or not server-based content has been locked.

Syntax

Boolean isLockedStoredContent(input String content_path)

Arguments

content_path URI of server-based content to be verified

Return values

true: The specified server-based content has been locked

false: The specified server-based content has not been locked

Description

This function verifies whether or not server-based content has been locked.

- deleteStoredContent(): Deletes server-based content.

Syntax

Number deleteStoredContent(input String content_path)

Arguments

content_path URI of server-based content to be deleted

Return values

1: Success

-1: Parameter error

-820: The specified server-based content has not been found

-823: The specified server-based content has been locked

-825: The specified server-based content is being played back

-826: The specified server-based content is being stored

NaN: Failure due to other causes

Description

This function deletes the server-based content designated by content_path. An explicit instruction or permission of an end user should be obtained to execute this function.

- exportContent(): Exports resources contained in server-based content.

Syntax

Array exportContent(
 input Array resource_path,
 input Number type
)

Arguments

resource	path[0]	Resource path
----------	---------	---------------

A necessary number of resource path arguments follow.

type	Process type
------	--------------

Return values

Array containing the result: Success

Array[0]:Indicates whether or not a CAS client properly works

0:Normal termination

-813:No external output functions are available

-820: The specified server-based content has not been found

-860:No CAS clients are available

-861:Found CAS clients are invalid

-862:Error on a CAS client

-881:No license for storing into a receiver, as specified

NaN:Failure due to other causes

Array[1]:Result of what has been executed via a CAS client

null: Failure

Description

This function exports resources designated by `resource_path` to the destination configured by the receiver. This function accepts paths of resources contained in server-based content as the argument `resource_path`.

7.6.18.6 Playback control functions

- `launchContent()`: Makes a transition to specified server-based content (resource) and starts to replay the server-based content (resource).

Syntax

```
Array launchContent(
    input String content_path,
    input String license_id
    [,input String ret_content_path]
)
```

Arguments

content_path	URI of server-based content (resource) to which a transition is to be made
--------------	--

license id	License ID
------------	------------

ret_content_path	URI of server-based content (resource) to which the second transition is to be made
------------------	---

Return values

Array containing the result:Success

Array[0]: Indicates whether or not a CAS client properly works

1:Success

-1:Parameter error
-820: The specified server-based content has not been found
-860:No CAS clients are available
-862:Error on a CAS client
-884:No licenses for the specified server-based content are available
NaN:Failure due to other causes
Array[1]:Result of what has been executed via a CAS client
null: Failure

Description

This function starts to play back the specified server-based content or resource. This function accepts a URI of server-based content or resource to be played back as the argument `content_path`. When there are two or more licenses for the server-based content designated by `content_path`, the argument `license_id` can be used to specify a license ID. Details on results of what is executed via a CAS card are defined in an operational standard regulation. The argument `ret_content_path` can be used to specify server-based content or resource that is played back immediately after a playback of server-based content or resource designated by `content_path` ends.

- `playSegment()`: Starts to play back an AV resource scene identified by a specified segment or group of segments.

Syntax

```
Array playSegment(  
    input String segment_ref  
    [,input String ret_content_path]  
)
```

Arguments

<code>segment_ref</code>	Specifies a segment or a group of segments of a server-based content that has been stored
<code>ret_content_path</code>	URI of the server-based content

Return values

Array containing the result: Success
Array[0]:Indicates whether or not a CAS client properly works
-1:Parameter error
-841:The specified segment or group of segments has not been found
-842:The video scene identified by the specified segment or group of segments has not been stored
-860:No CAS clients are available
-862:Error on a CAS client
-883:No server-based content containing the video scene identified by the specified segment or group of segments has been found
NaN:Failure due to other causes

Array[1]:Result of what has been executed via a CAS client

null: Failure

Description

This function starts to replay a video scene identified by a segment or group of segments designated by `segment_ref` and then quits a data broadcasting engine. When the playback of the specified scene ends, the function activates a resource designated by `ret_content_path`. The description of `segment_ref` conforms to the namespace conventions defined in Chapter 9. Details on how the result of what is executed via a CAS client is treated are defined in an operational standard regulation.

- `launchDynamicDocument()`: Presents a BML document generated dynamically by a communication server.

Syntax

```
Number launchDynamicDocument(  
    input String src_path,  
    input String post_body  
)
```

Arguments

`src_path` URI of a communication server that generates a BML document dynamically
`post_body` Information that accompanies an http request to a communication server

Return values

1: Success
NaN: Failure

Description

The function accepts strings designated by `post_body` as the BODY information for the POST method and sends the information with an http request to a communication server. The receiving communication server uses the BODY information to generate a BML document dynamically and then returns the BML document to the browser. The browser, in turn, presents the returned BML document.

7.6.18.7 Metadata reference functions

- `getMetadataElement()`: Obtains metadata elements that are stored in a receiver.

Syntax

```
string getMetadataElement(  
    input Number scope,  
    input Array node_info,  
    input Number order  
)
```

Arguments

`scope` Identifies a storage area in the receiver
`node_info` Identifies nodes
`order` Specifies an order of nodes

Return values

Strings describing elements: Success
null: Failure

Description

This function obtains strings contained in metadata elements stored in the receiver, as uniquely identified in arguments. When no metadata elements are found based on the values in arguments, the function returns null. This function accepts a number that identifies a storage area in the receiver as the argument scope. This function accepts an Array object that has information to uniquely identify a metadata element in the receiver as the argument node_info. Details on the description of node_info are defined in an operational standard regulation. The argument order is used to specify an order in which two or more metadata elements appear, when two or more metadata elements share a name.

- getSynopsis(): Obtains metadata elements in the receiver that provide a Synopsis.

Syntax

```
Array getSynopsis(  
    input Number scope,  
    input Array node_info  
)
```

Arguments

scope Identifies a storage area in a receiver
node_info Identifies nodes

Return values

Array containing a Synopsis: Success
Array[0]: Result of the execution
Array[1] and following elements: Provide a Synopsis
Details on arrangements of resulting arrays and allocations of metadata elements to arrays are defined in an operational standard regulation.
null: Failure

Description

This function obtains metadata elements that provide a Synopsis and returns them in an Array object (For more information on Synopsis, refer to Chapter 3, ARIB-STD-B38). This function accepts a number that identifies a storage area in the receiver as the argument scope. This function accepts an Array object that has information to uniquely identify a metadata element in a receiver as the argument node_info. Details on the description of node_info are defined in an operational standard regulation.

- searchMetadata(): Obtains a list of metadata elements that are stored in a receiver and satisfy conditions.

Syntax

```
Array searchMetadata(  
    input String scope,
```

```
input String node,  
input Number from,  
input Number count,  
input Array condition  
)
```

Arguments

scope Identifies storage areas to be searched

nodeIdentifies nodes

fromSpecifies a place in an order of the found metadata elements to put the metadata element in the place into Array[1]

count Specifies a number of metadata elements to be returned

condition Arrays containing search conditions

Return values

List of searched elements: Success

Array[0][0]:Result of the execution including the number of found metadata elements

-840:No metadata elements that satisfy conditions have been found

0:Indicates that the number of found metadata elements is less than the number designated by from

1 or greater: Represents the number of found metadata elements

Array[1] and following elements: Contain found metadata elements

null: Failure

Description

This function searches storage areas in the receiver for metadata elements, as specified in arguments including the condition argument that provides search conditions, to obtain a list of found metadata elements. The first item of the returned list is a metadata element of which place in the order of the list of the found metadata elements is equivalent to a value in the from argument. The number of returned metadata elements is equivalent to a value in the count argument. This function accepts a number that identifies a storage area in the receiver as the argument scope. Details on the description of scope are defined in an operational standard regulation. This function accepts strings that have information to uniquely identify a metadata element in the receiver as the argument node. Details on relationships between nodes and strings used to identify nodes are defined in an operational standard regulation. A value in the from argument represents a place in an order of the found metadata elements to put the metadata element in the place into Array[1]. A value in the count argument represents a number of metadata elements to be returned, with the first returned element designated by from. The argument condition contains an Array object that provides search conditions including descriptions of wanted metadata elements and key strings. Details on relationships between descriptions of wanted metadata elements and key strings are defined in an operational standard regulation. A returned Array object indicates what has resulted from the execution of this function including the number of found metadata elements and a selected list of found metadata elements. Details on arrangements of the elements in the resulting array and allocations of found metadata elements to the elements of the resulting array are defined in an operational standard regulation.

- searchMetadataOnServer(): Obtains a list of metadata elements that are stored in a communication server and satisfy conditions.

Syntax

```
Array searchMetadataOnServer(  
    input String uri,  
    input String node,  
    input Number from,  
    input Number count,  
    input Number sort,  
    input Number type,  
    input String condition  
)
```

Arguments

uri URI of a communication server

nodeIdentifies nodes

fromSpecifies a place in an order of the found metadata elements to put the metadata element in the place into Array[1]

count Specifies a number of metadata elements to be returned

sort Specifies how to sort the result

type Specifies how the communication server responds to this function

condition Arrays containing search conditions

Return values

List of searched elements: Success

Array[0][0]:Result of the execution

-840:No metadata elements that satisfy search conditions have been found

0:Indicates that the number of found metadata elements is less than the number designated by from

1 or greater:Represents the number of found metadata elements

Array[1] and following elements: Contain found metadata elements

null: Failure

Description

This function searches storage areas in a communication server for metadata elements, as specified in the condition argument that provides search conditions, to obtain metadata elements that satisfy the search conditions and a list of the found metadata elements. This function accepts a URI of a communication server to be searched as the uri argument. This function accepts strings that have information to uniquely identify a metadata element in the receiver as the argument node. Details on relationships between nodes and strings used to identify nodes are defined in an operational standard regulation. A value in the from argument represents a place in an order of the found metadata elements to put the metadata element in the place into Array[1]. A value in the count argument represents a number of metadata elements to be returned, with the first returned element designated by from. The argument sort specifies how to sort the result. Details on relationships between values and sort methods are defined in an operational standard regulation. The argument type specifies how the communication server responds to this function (for example, the communication server gives only a list of found metadata elements or sends metadata elements that have been found). Details on relationships between values and response methods are defined in an operational standard regulation. The argument condition contains an Array object that provides search conditions including descriptions

of wanted metadata elements and key strings. Details on relationships between descriptions of wanted metadata elements and key strings are defined in an operational standard regulation. A returned Array object indicates what has resulted from the execution of this function including the number of found metadata elements and a selected list of found metadata elements. Details on arrangements of the elements in the resulting array and allocations of metadata elements to the elements of the resulting array are defined in an operational standard regulation.

- `getEntryResourceInformation()`: Obtains an Array object that contains entry resource information of server-based content.

Syntax

Array `getEntryResourceInformation`(input String `content_path`)

Arguments

`content_path` URI of server-based content that has been stored

Return values

Array containing information: Success

null: Failure to obtain intended information

Description

This function obtains an Array object that contains the entry resource information of the server based content specified by the URI in argument `content_path`. Details on arrangements of arrays containing information and allocations of information elements to arrays are defined in an operational standard regulation.

7.6.18.8 Communication functions

- `downloadContent()`: Obtains server-based content and stores it in a receiver.

Syntax

Number `downloadContent`(input String `src_path`)

Arguments

`src_path` URI of server-based content on a communication server to be downloaded

Return values

1: Success (The storing process has started)

NaN: Failure

Description

This function starts to store server-based content on a communication server, as specified in `src_path`.

- `downloadResources()`: Identifies resources in a communication server, obtains them, and stores them in a receiver.

Syntax

Number `downloadResources` (
 input String `src_path`,
 input Array `resource_path`,

```
    input String dest_path,  
    input Number update_type  
)
```

Arguments

src_path URI of server-based content on a communication server to be downloaded

resource_path Array identifying resources on a communication server to be downloaded

Array[0] and following elements: Resource URIs

dest_path URI in which obtained resources are to be stored

update_type Flag indicating whether or not obtained resources accept updates

Return values

1: Success (The storing process has started)

NaN: Failure

Description

This function obtains resources that have been stored in a communication server, as specified in src_path and resource_path, and stores them under a directory designated by dest_path.

- testNetwork(): Measures a period of time in which a specified downloading of resources in a communications occurred.

Syntax

Array testNetwork(input String src_path)

Arguments

src_path URI of a resource stored in a communication server

Return values

Arrays containing information indicating what has resulted from the execution:

Array[0]:Contains a number representing the result

1:Failure

-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

-602:Invalid file name

-603:Not enough storage space available on the storage device

-700:Service was disconnected

NaN:Failure due to other causes

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

Array[2]:Length in bytes of a received file

Array[3]:Media type described in the Content-Type header in HTTP1.1

Array[4]:Length of the period of time in which the downloading occurred

Description

This function downloads resources stored in an HTTP server, as specified in `src_path`, and returns a value that represents the length of the period of time that the downloading required. Note that this function does not store the downloaded resources into the receiver. This function returns an Array object to indicate what resulted from the execution, including errors during the connection, a Status-Code in HTTP1.1, the number of received files, and the Media type. A returned value containing a length in bytes of a received file represents a value in the Content-Length header in HTTP.

- `getHttpResponseHeader()`: Obtains an Array object containing HTTP response information about a communication session to a communication server.

Syntax

Array `getHttpResponseHeader(input String uri)`

Arguments

uri URI of an HTTP server

Return values

Arrays containing information indicating what has resulted from the execution:

Array[0]:Contains a number representing the result

1:Failure

-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

-700:Service was disconnected

NaN:Failure due to other causes

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

Array[2]:Length in bytes of a received file

Description

This function accesses the communication server designated by `uri` and obtains a HTTP response header. The maximum length of a HTTP response header to be obtained is defined in an operational standard regulation.

- `setServerInfo()`: Registers information identifying a communication server.

Syntax

```
Number setServerInfo(
    input String broadcaster,
    input String server_url,
    input Number server_type,
    input Number type
)
```

Arguments

broadcaster	Broadcaster description
server_url	URL of a communication server
server_type	Type of a communication server
type	Type of a status to be registered

Return values

1: Normal termination
 -1: Parameter error
 -812: No storage area available to store the information
 NaN: Abnormal termination due to other causes

Description

This function registers information that identifies a communication server specific to the server-based broadcasting, including the URL and the server type, in a non-volatile memory in the receiver. The registered information remains effective in establishing a connection to the communication server until the information is overwritten by this function or a receiver resident function. The argument broadcaster contains a string that uniquely identifies a broadcaster. The argument server_type specifies a function type of a communication server (including a communication server for CAS and a communication server for metadata). Details on the arguments type, broadcaster, and server_type are defined in an operational standard regulation.

- getServerInfo(): Obtains information on communication servers.

Syntax

Array getServerInfo(input String broadcaster)

Arguments

broadcaster	Broadcaster description
-------------	-------------------------

Return values

Array containing registered information: Success
 null: Failure

Description

This function returns an Array object containing information on a communication server specific to the server-based broadcasting that has been registered in a non-volatile memory in the receiver. Details on arrangements and values of returned arrays containing information are defined in an operational standard regulation.

7.6.18.9 Other functions

- getStorableSpace(): Obtains a value indicating storage capacity available for storing content in a

receiver.

Syntax

Number getStorableSpace()

Arguments

None

Return values

Value indicating available storage capacity: Success

NaN: Failure

Description

This function returns a value that indicates storage capacity available for storing content in a storage device in a receiver in megabytes (MB), assuming that 1 MB equals 1,000,000 bytes. This function returns the value 0, when available storage capacity is less than 1 MB. Any returned value is intended to be informational only and may not represent a strictly accurate value.



- setTune(): Reserves a tuner for a specified service.

Syntax

Number setTune(input String service_ref)

Arguments

server_ref A service for which a tuner is to be reserved

Return values

1: Success

NaN: Failure

Description

The description of server_ref conforms to the namespace conventions defined in Chapter 9. This function fails, when a tuner in a receiver is occupied by other processes including a storing process, being not available for a specified service. Any other process has a higher priority over a reservation done by this function; a tuner in a receiver that has been reserved for a service by this function accepts another process including a storing process.

- getDirStructures(): Obtains directory names and information indicating whether or not child directories of each directory have been found.

Syntax

Array getDirStructures(input String path)

Arguments

path URI identifying a (parent) directory

Return values

Array containing strings of (child) directory names and values indicating whether or not (grandchild) directories have been found: Success

Array[0]: (child) directory names and values indicating whether or not (grandchild) directories have been found

Array[0][0]:(child) directory name

Array[0][1]:value indicating whether or not (grandchild) directories have been found

1:(Grandchild) directories have been found, 0:No (grandchild) directories have been found

More elements follow,as necessary

null: Failure (No specified path has been found)

Description

This function returns names of directories (child directories) of a specified directory (a parent directory). This function also returns a value indicating whether or not directories (grandchild directories of a specified directory) under each of found directory (a child directory of a specified directory) have been found. The value 1 indicates that directories (grandchild directories of a specified directory) immediately under each of found directories (child directories of a specified directory) have been found. The value 0 indicates that directories (grandchild directories of a specified directory) immediately under each of found directories (child directories of a specified directory) have not been found. All the returned information is contained in an Array object. When no directories (child directories) under a specified directory (a parent directory) have been found, an array of which length is 0 is returned.

- getTransitSource(): Verifies which source invoked the currently executed content.

Syntax

Array getTransitSource()

Arguments

None

Return values

Information identifying an invoking source:

Array[0]:Identification of an invoking source

Array[1]:Details on an invoking source

Description

This function verifies which source invoked the currently executed content. Details on values that this function returns, scope relationships between an invoking source and the currently executed content are defined in an operational standard regulation.

7.7 Navigator pseudo object properties

- appName: Navigator Pseudo Object Properties

Syntax

navigator.appName

Description

A string identifying a BML browser. The available values are defined in an operational standard regulation.

- appVersion: Navigator Pseudo Object Properties

Syntax

navigator.appVersion

Description

A string representing a version of a BML browser. The available values are defined in a receiver.

7.8 Functions for interoperability with JavaScript

The security class applicable to each of the following functions complies with the security class for the corresponding extended function for broadcasting.

- Location.reload()

Syntax

location.reload()

Description

This function is an equivalent to reloadActiveDocument().

- location.replace()

Syntax

location.replace()

Description

This function is an equivalent to launchDocument().

7.9 Security Class for content and Extended Functions for Broadcasting

Each extended function for broadcasting defined in Section 7.6 is applicable to Class A content/Class B content defined in Section 7.3 as shown in the following Table 7-6.

Table 7-6 Applicability of Extended Function for Broadcasting to Security Class

API	Class A	Class B
EPG functions (7.6.1)		
epgGetEventStartTime()	O	O
epgGetEventDuration()	O	O
epgTune()	O	O
epgTuneToComponent()	O	O
epgTuneToDocument()	O	O
epgIsReserved()	O	O
epgReserve()	O	O
epgCancelReservation()	O	O
epgRecIsReserved()	O	O
epgRecReserve()	O	O *
epgRecCancelReservation()	O	O *
Event group index functions (7.6.2)		
grpIsReserved()	O	O
grpReserve()	O	O

API	Class A	Class B
grpCancelReservation()	O	O
grpRecIsReserved()	O	O *
grpRecReserve()	O	O *
grpRecCancelReservation()	O	O
grpGetNodeEventList()	O	O
grpGetERTNodeName()	O	O
grpGetERTNodeDescription()	O	O
epgXTune()	O	O
Series reservation functions (7.6.3)		
seriesIsReserved()	O	O
seriesReserve()	O	O
seriesCancelReservation()	O	O
seriesRecIsReserved()	O	O
seriesRecReserve()	O	O *
seriesRecCancelReservation()	O	O *
Subtitle presentation control functions (7.6.4)		
setCCStreamReference()	O	-
getCCStreamReference()	O	-
setCCDisplayStatus()	O	-
getCCDisplayStatus()	O	-
getCCLanguageStatus()	O	-
Non-volatile memory functions (7.6.5)		
Non-volatile memory functions - Functions for controlling non-access-controlled areas (7.6.5.1)		
writePersistentString()	O	-
writePersistentNumber()	O	-
writePersistentArray()	O	-
readPersistentString()	O	-
readPersistentNumber()	O	-
readPersistentArray()	O	-
copyPersistent()	O	-
getPersistentInfoList()	O	-
deletePersistent()	O	-
getFreeSpace()	O	-
Non-volatile memory functions - Functions for controlling access-controlled areas (7.6.5.2)		
setAccessInfoOfPersistentArray()	O	-
checkAccessInfoOfPersistentArray()	O	-
writePersistentArrayWithAccessCheck()	O	-
readPersistentArrayWithAccessCheck()	O	-
Extended APIs for Storing (7.6.6)		
Extended APIs for Storing - Directory Management Functions (7.6.6.1)		
saveDirAs()	O	-
saveDir()	O	-
createDir()	O	-
getParentDirName()	O	-
getDirNames()	O	O
isDirExisting()	O	O
Extended APIs for Storing - File Management Functions (7.6.6.2)		
saveFileAs()	O	-
saveFile()	O	-
getFileNames()	O	O
isFileExisting()	O	O

API	Class A	Class B
Extended APIs for Storing - File Input/Output Functions (7.6.6.3)		
writeArray()	O	-
readArray()	O	-
Extended APIs for Storing - Inquiry Functions (7.6.6.4)		
getDirInfo()	O	-
getFileInfo()	O	-
getContentSource()	O	-
getStorageInfo()	O	-
getCarouselInfo()	O	-
getModuleInfo()	O	-
Extended APIs for Storing - Data Carousel Storage Function (7.6.6.5)		
saveCarouselAs()	O	-
saveCarousel()	O	-
saveModuleAs()	O	-
saveModule()	O	-
saveResourceAs()	O	-
saveResource()	O	-
Interaction Channel functions (7.6.7)		
Interaction Channel functions - Communication Functions assuming simple protocols including BASIC procedures (7.6.7.1)		
connect()	O	-
disconnect()	O	-
sendBinaryData()	O	-
receiveBinaryData()	O	-
sendTextData()	O	-
receiveTextData()	O	-
Interaction Channel functions - Delayed call functions assuming simple protocols including BASIC procedures (7.6.7.2)		
registerTransmission()	O	-
registerTransmissionStatus()	O	-
getTransmissionStatus()	O	-
setDelayedTransmissionDataOverBASIC()	O	-
Interaction Channel functions - Communication functions using the mass calls reception service (7.6.7.3)		
vote()	O	-
Interaction Channel functions - Functions for encrypted communication using CAS (7.6.7.4)		
startCASEncryption()	O	-
endCASEncryption()	O	-
transmitWithCASEncryption()	O	-
Interaction Channel functions - Functions for communication with public key encryption not using CAS (7.6.7.5)		
setEncryptionKey()	O	-
beginEncryption()	O	-
endEncryption()	O	-
Interaction Channel functions - Communication functions assuming TCP/IP (7.6.7.6)		
setISPParams()	O	-
getISPParams()	O	-
connectPPP()	O	-
connectPPPWithISPParams()	O	-
disconnectPPP()	O	-
getConnectionType()	O	O

API	Class A	Class B
isIPConnected()	O	O
saveHttpServerFileAs()	O	-
saveHttpServerFile()	O	O *
sendHttpServerFileAs()	O	-
saveFtpServerFileAs()	O	-
saveFtpServerFile()	O	-
sendFtpServerFileAs()	O	-
sendTextMail()	O	-
transmitTextDataOverIP()	O	-
setDelayedTransmissionData()	O	-
setCacheResourceOverIP()	O	O
Interaction Channel functions - Status look-up functions for delayed call functions applicable to BASIC procedures and IP connections (7.6.7.7)		
getDelayedTransmissionStatus()	O	-
getDelayedTransmissionResult()	O	-
Interaction Channel functions - Function for obtaining line connection status (7.6.7.8)		
getPrefixNumber()	O	-
Functions for operating root certificates for encrypted transmission (7.6.7.9)		
isRootCertificateExisting()	O	O
getRootCertificateInfo()	O	O
Operational control functions (7.6.8)		
reloadActiveDocument()	O	O
getNPT()	O	O
getProgramRelativeTime()	O	O
isBeingBroadcast()	O	O
lockExecution()	O	-
unlockExecution()	O	-
lockModuleOnMemory()	O	-
unlockModuleOnMemory()	O	-
setCachePriority()	O	O
getTuningLinkageSource()	O	-
getTuningLinkageType()	O	-
getLinkSourceServiceStr()	O	-
getLinkSourceEventStr()	O	-
getIRDID()	O	-
getBrowserVersion()	O	O
getProgramID()	O	O
getActiveDocument()	O	O
lockScreen()	O	O
unlockScreen()	O	O
getBrowserSupport()	O	O
launchDocument()	O	O
launchDocumentRestricted()	O	-
quitDocument()	O	O
launchExApp()	O	O
getFreeContentsMemory()	O	O
isSupportedMedia()	O	O
detectComponent()	O	O
lockModuleOnMemoryEx()	O	-
unlockModuleOnMemoryEx()	O	-

API	Class A	Class B
unlockAllModulesOnMemory()	O	-
getLockedModuleInfo()	O	-
setFullDataDisplayArea()	O	O
getDataDisplayAreaSize()	O	O
getResidentAppVersion()	O	O
startResidentApp()	O	O
Receiver sound control (7.6.9)		
playRomSound()	O	O
Timer functions (7.6.10)		
sleep()	O	O
setTimeout()	O	O
setInterval()	O	O
clearTimer()	O	O
pauseTimer()	O	O
resumeTimer()	O	O
setCurrentDateMode()	O	O
External character functions (7.6.11)		
loadDRCS()	O	O
unloadDRCS()	O	O
Functions for controlling external devices (7.6.12)		
enumPeripherals()	O	O
passXMLDocToPeripheral()	O	O
getArrayFromPeripheral()	O	O
Functions for controlling bookmark areas (7.6.13)		
writeBookmarkArray()	O	-
readBookmarkArray()	O	O
deleteBookmark()	O	-
lockBookmark()	O	-
unlockBookmark()	O	-
getBookmarkInfo()	O	O
Other functions (7.6.14)		
random()	O	O
subDate()	O	O
addDate()	O	O
formatNumber()	O	O
Ureg pseudo object properties (7.6.15)		
Ureg[]	O	-
Greg pseudo object properties (7.6.16)		
Greg[]	O	O
Functions for printing (7.6.17)		
getPrinterStatus()	O	O
printFile()	O	-
printTemplate()	O	-
printUri()	O	O
printStaticScreen()	O	O
saveImageToMemoryCard()	O	-
saveStaticScreenToMemoryCard()	O	O
saveHttpServerImageToMemoryCard()	O	O
Server-based broadcasting functions (7.6.18)		
Storage schedule functions (7.6.18.1)		

API	Class A	Class B
epgStoreReserve()	O	O Note
epgStoreCancelReservation()	O	O Note
epgStoreCheckReservation()	O	O
seriesStoreReserve()	O	O Note
seriesStoreCancelReservation()	O	O Note
seriesStoreCheckReservation()	O	O
Storage functions (7.6.18.2)		
storeStart()	O	O Note
storeTerminate()	O	O Note
checkStoreStatus()	O	O
License functions (7.6.18.3)		
getLicense()	O	-
getLicenseIDList()	O	-
getLicenseStatus()	O	-
setPurchaseInfo()	O	-
getPurchaseInfo()	O	-
getLicenseLinkInfo()	O	-
getEncryptionkeyInfo()	O	-
CAS functions (7.6.18.4)		
requestCASProcess()	O	-
getCASInfo()	O	-
Server-based content control functions (7.6.18.5)		
isContentStored()	O	-
lockStoredContent()	O	-
unlockStoredContent()	O	-
isLockedStoredContent()	O	-
deleteStoredContent()	O	-
exportContent()	O	-
Playback control functions (7.6.18.6)		
launchContent()	O	O
playSegment()	O	O
launchDynamicDocument()	O	O
Metadata reference functions (7.6.18.7)		
getMetadataElement()	O	O
getSynopsis()	O	O
searchMetadata()	O	O
searchMetadataOnServer()	O	O
getEntryResourceInformation()	O	O
Communication functions (7.6.18.8)		
downloadContent()	O	O Note
downloadResources()	O	O Note
testNetwork()	O	O
getHttpResponseHeader()	O	O
setServerInfo()	O	-
getServerInfo()	O	-
Other functions (7.6.18.9)		
getStorableSpace()	O	O
setTune()	O	-
getDirStructures()	O	O
getTransitSource()	O	O

Legend) O: Applicable, -: Not applicable

* : The applicability must complies with the guidelines on recording and storing defined in Appendix 1.

Table 7-7 Applicability of Navigator pseudo objects to Security Class

API	Class A	Class B
appName	O	O
appVersion	O	O

Chapter 8 Monomedia Coding Schemes and Transmission Used in BML/B-XML Documents

This chapter defines coding schemes and its transmission methods for the monomedia data used in BML and B-XML documents. It is assumed that encoding schemes and transmission methods that are not defined in this chapter may be applied in real operations of each transmission media type. Therefore a BML browser must successfully ignore any coding scheme or transmission method that is not supported by the browser not to affect any normal operation.

8.1 Video Coding Scheme and Transmission

The video coding scheme and transmission method defined in this section are applied to video data that is referenced by object element.

8.1.1 Transmission of MPEG-1 video

8.1.1.1 Transmission in video PES

To transmit MPEG-1 Video data through video PES in MPEG2-TS, it must be transmitted as a stream with the stream type value of 0x01.

8.1.1.2 Transmission in data carousel

To transmit MPEG-1 Video data through a data carousel (stream type value of 0x0B or 0x0D), it must be transmitted in following way

- 1) As a file of multiplexed stream in MPEG-1 systems.
- 2) In a multiplexed, time-stamped TS format , as specified in Section 8.1.4.

8.1.2 Transmission of MPEG-2 video

8.1.2.1 Transmission in video PES

To transmit MPEG-2 Video data through video PES in MPEG2-TS, it must be transmitted as a stream with the stream type value of 0x02.

8.1.2.2 Transmission in data carousel

To transmit MPEG-2 Video data through a data carouse (stream type value of 0x0B or 0x0D), it must be multiplexed into a time-stamped TS format specified in Section 8.1.4.

8.1.3 Transmission of MPEG-4 video and H.264|MPEG-4 AVC

8.1.3.1 Transmission in video PES

To transmit MPEG-4 Video data through video PES in MPEG2-TS, it must be transmitted as a stream with the stream type value of 0x10.

To transmit H.264|MPEG-4 AVC data through PES in MPEG2-TS, it must be transmitted as a stream with the stream type value of 0x1B.

8.1.3.2 Transmission in data carousel

To transmit MPEG-4 Video or H.264|MPEG-4 AVC data through a data carousel (stream type value of 0x0B or 0x0D), it must be multiplexed into a time-stamped TS format specified in Section 8.1.4.

8.1.4 Transmission of MPEG video/audio in a time-stamped TS format

8.1.4.1 Data Encoding Specification

To transmit MPEG-1/2/4 Video or H.264|MPEG-4 AVC data along with MPEG-2/4 Audio data in multiplexed files in a data carousel, each multiplexed video/audio files are coded in a time-stamped TS format defined in Table 8-1.

Table 8-1 Time-stamped TS Format

Syntax	Bits	Mnemonic
<pre>TimeStampedPartialTS(){ do { Timestamp transport_packet() } while (! end_of_file) }</pre>	32	uimbsbf

timestamp: This 32-bit field contains a counter value of a 27 MHz clock synchronized with the MPEG system clock to control a relative time entered into a decoder for a transport packet. This field is not required to be associated with a STC counter value.

transport_packet(): This field represents a transport packet defined in ISO/IEC13818-1. A transport packet which contains PAT and PMT, also required SIT, video streams and audio streams are transmitted.
For actual operation, two different types of transmitting, transmitting at a variable bit rate and transmitting at a constant bit rate are assumed. To transmit at a variable bit rate, the packets required to play a concerned stream, including null packets, can be omitted from a transmitted stream to maximize the transmitting efficiency. To transmit at a fixed bit rate, null packets should be added to implement the concerned fixed bit rate.

8.1.4.2 Required Tables for PSI

- PAT

A PAT must be described.

Any PAT must be described with program_number whose value is other than 0 and must represent PID of PMT. The available values of program_number are defined in an operational standard regulation.

- PMT

A PMT must be described.

Any stream identification descriptor indicating a second loop must contain a PMT descriptor. Otherwise, a descriptor may be placed as required.

It is recommended the available values to component_tag, and the occurrence rules of component_tag in a default ES and PMT descriptors in a second loop are equivalent to an

operational standard regulation used for the main stream of the media type responsible for transmitting the concerned stream.

- SIT

In an implementation in which a transport stream that is decoded from a file that has been transmitted based on the data encoding specification defined in Section 8.1.4.1 is presented in a high-speed digital interface, an SIT must be described, as specified in ARIB STD-B21.

In other cases, SITs are not required unless otherwise specified explicitly.

- Other considerations

- Any table other than PAT, PMT, and SIT (e.g. CAT, NIT, SDT, BAT, EIT, RST, TDT, TOT, PCAT, SDTT, ST) must not be described.
- A PAT must occur in a stream at a frequency of not less than one time per 100 milliseconds. A PMT must occur in a stream at a frequency of not less than one time per 100 milliseconds.
- As far as in a single time-stamped TS format file, a PAT/PMT must not be modified nor updated.

8.1.4.3 Transmission in data carousel

To transmit a file coded with the data encoding specification defined in Section 8.1.4.1 in a data carousel, the transmission must comply with the following.

- Alike general file, the file is divided into DDB whose lengths are defined in an operational standard regulation from the beginning.
- A time-stamped TS descriptor defined below must be contained in DII.

Table 8-2 Time-stamped TS descriptor

Syntax	Bits	Mnemonic
TimestampedTSdescriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
TSbitrate	32	uimsbf
Reserved	3	bslbf
PCR_PID	13	uimsbf
default_video_flag	1	bslbf
default_audio_flag	1	bslbf
timestampedTS_PBctrl_info_flag	1	bslbf
Reserved	5	bslbf
if (default_video_flag == 0) {		
reserved	3	bslbf
videoPID	13	uimsbf
}		
if (default_audio_flag == 0) {		
reserved	3	bslbf
audioPID	13	uimsbf
}		
if (partialTS_PBctrl_info_flag == 0) {		
Text_length	8	uimsbf
for (i=0; i<text_length; i++) {		
text_ch	8	uimsbf
}		
}		
}		

TSbitrate (TS bit rate):

This field contains a value representing a bit rate of the concerned transport stream in 1 Hz units. When a variable bit rate is used, the maximum value is contained.

PCR_PID:

This 13-bit field contains PID of a stream transmitting PCR used to play the transport stream.

default_video_flag (video flag): This field contains a flag indicating an existence of video streams that must be played by default during a play of the time-stamped TS file.

- 0: A video stream that must be played by default exists. Their PIDs are specified.
- 1: No video stream that must be played by default exists.

default_audio_flag (audio flag):

This field contains a flag indicating an existence of audio streams that must be played by default during a play of the time-stamped TS file.

- 0: An audio stream that must be played by default exists. Their PIDs are specified.
- 1: No audio stream that must be played by default exists.

timestampedTS_PBctrl_info_flag (flag of information to control playback of time-stamped TS file):

This field contains a flag indicating an existence of information to control playing the time-stamped TS file, including configurations for special playbacks and instant accesses to desired locations.

- 0: Information to control playing partial TS exist. Its URI is specified.
- 1: No information to control playing partial TS exists.

videoPID (video PID):

This 13-bit field contains PID of a video stream to be played by default.

audioPID (audio PID):

This 13-bit field contains PID of an audio stream to be played by default.

text_length (string length):

This 8-bit field contains the length of a URI string described in the following field.

text_ch (URI character codes):

This 8-bit field contains a URI string locating information to control playing the time-stamped TS file. Any detailed specification of information to control playing the time-stamped TS file is defined in a standard other than this standard.

8.1.4.4 Constraints in playing

To perform receiving a broadcasting service and playing a time-stamped TS file at the same time, two separate transport stream processing systems are required. The constraints in integrating and coordinating a content/event message received via a broadcasting service and a time-stamped TS file are described in a standard other than this standard.

8.2 Coding Scheme and Transmission of Still Pictures and Bitmap Graphics

The coding schemes and transmission methods defined in this section are applied to still pictures and bitmap graphics data referenced by object element as well as still picture data referenced by background-image attribute.

8.2.1 Transmission of MPEG-2 I-frame, MPEG-4 I-VOP, and H.264|MPEG-4 AVC I-picture

8.2.1.1 Transmission in video PES for linear playback

To transmit a still picture in MPEG-2 I-frames through a video PES component, the coding scheme must conform to the conventions defined in ARIB STD-B24 Volume 1, Part 2, 5.1. The PES component must be transmitted as a stream with the stream type value of 0x02.

To transmit a still picture in MPEG-4 I-VOP through a video PES component, the coding scheme must conform to the conventions defined in ARIB STD-B24 Volume 1, Part 2, 5.1.2. The PES component must be transmitted as a stream with the stream type value of 0x10.

To transmit a still picture in H.264|MPEG-4 AVC I-picture through a video PES component, the coding scheme must conform to the conventions defined in ARIB STD-B24 Volume 1, Part 2, 5.1.3. The PES component must be transmitted as a stream with the stream type value of 0x1B.

8.2.1.2 Transmission in carousel module for interactive playback

To transmit a still picture in MPEG-2 I frames in a carousel module, the coding scheme must conform to the conventions in ARIB STD-B24 Volume 1, Part 2, 5.1. The still picture must be transmitted as a file in the module.

To transmit a still picture in MPEG4-I-VOP in a carousel module, the coding scheme must conform to the conventions in ARIB STD-B24 Volume 1, Part 2, 5.1.2. The still picture must be transmitted as a file in the module.

To transmit a still picture in H.264|MPEG-4 AVC I-picture in a carousel module, the coding scheme must conform to the conventions in ARIB STD-B24 Volume 1, Part 2, 5.1.3. The still picture must be transmitted as a file in the module.

In these cases, the stream type value of the data carousel component must be 0x0B or 0x0D.

8.2.1.3 Transmission in video PES for interactive playback (Still Picture Carousel)

This transmission method is designed to enable decoding still pictures by locating the content only on the memory of a video decoder, when displaying MPEG2 I-frame, MPEG4-I-VOP, or H.264|MPEG-4 AVC I-picture still pictures interactively from data broadcasting applications.

- Multiplexing of and Conditions for a Still Picture Carousel

The following two components are used for transmitting a still picture carousel.

- (1) Still Picture Component: This component is a Video PES component for transmitting an entity of I-frame, I-VOP, or I-picture still picture data (stream type value is 0x02 or 0x10). Each component may contain up to 16 video PESes. Each video PES has a unique stream_id within the Still Picture Components. It transmits one or more I-frame, I-VOP, or I-picture still pictures whose coding scheme must conform to the specifications defined in ARIB STD-24 Volume 1, Part 2, 5.1.1 "MPEG-I frame", 5.1.2 "MPEG-4 I-VOP", of 5.1.3 "H.264|MPEG-4 AVC I-picture".

- (2) IIT Component: This component is used for transmitting I-frame Identifying Table(IIT), which is a section format conforming to the DDB format of data carousel. The IIT retains the relationship between I-frame_ID, and video PES that is identified by a pair of component_tag and stream_id. An I-frame, I-VOP, or I-picture still picture specified with I-frame_ID is filtered by the MPEG decoder in the receiver with a combination of component_tag and stream_id by referencing IIT. This component must be specified as a component that composes the content.

There are following conditions for multiplexing I-frame still pictures and IITs.

- The IIT indicates the timing and the location where the I-frame, I-VOP, or I-picture still picture is multiplexed. It must appear at a certain time (e.g. 1 second³) before the corresponding I-frame, I-VOP, or I-picture starts.
 - Once an IIT that indicates the starting position of the I-frame, I-VOP, or I-picture has appeared, no other I-frame, I-VOP, nor I-picture must appear in a video PES where the specified I-frame, I-VOP, or I-picture belongs.
 - Once an I-frame, I-VOP, or I-picture has been multiplexed, no other I-frame, I-VOP, nor I-picture must appear in a video PES to which the specified I-frame, I-VOP, or I-picture belongs before the display of that I-frame, I-VOP, or I-picture is completed.
- IIT Encoding

Table 8-3 shows the structure of an I-frame Identifying Table. This structure conforms to the DDB format of data carousel.

Table 8-3 I-frame_Identifying_Section

Syntax	Number of bits	Mnemonic
Iframe_Identifying_section(){		
table_id = 0x3C	8	uimsbf
section_syntax_indicator = 1	1	bslbf
private_indicator = 0	1	bslbf
Reserved	2	bslbf
dsmcc_section_length	12	uimsbf
I-frame_id	16	uimsbf
Reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
protocolDiscriminator = 0x11	8	uimsbf
dsmccType = 0x03	8	uimsbf
messageID	16	uimsbf
downloadID	32	uimsbf
reserved = 0xff	8	uimsbf
adaptationLength = 0	8	uimsbf
messageLength = 0x0010	16	uimsbf
I-frame_id	16	uimsbf
I-frame_version	8	uimsbf
Reserved	8	bslbf
blockNumber	16	uimsbf
Reserved	4	bslbf

³ This timing may be changed to optimal value with respect to the performance of receivers. The actual value must be defined by the operational guideline.

first_pts[32..30]	3	bslbf
marker_bit	1	bslbf
first_pts[29..15]	15	bslbf
marker_bit	1	bslbf
first_pts[14..0]	15	bslbf
marker_bit	1	bslbf
Reserved	4	bslbf
last_pts[32..30]	3	bslbf
marker_bit	1	bslbf
last_pts[29..15]	15	bslbf
marker_bit	1	bslbf
last_pts[14..0]	15	bslbf
marker_bit	1	bslbf
stream_id	8	uimsbf
component_tag	8	uimsbf
CRC_32	32	rpchof
}		

- table_id :** An 8-bit field that indicates the type of the section. This field must be set to 0x3C according to the DSM-CC DDB.
- section_length :** A 12-bit field that indicates the number of bytes in the area from the beginning of the immediately following field to the end of CRC_32 field. The value of Section_length must not exceed 4093 so that the whole section does not exceed 4096.
- I-frame_id :** A 16-bit field that indicates an I-frame/I-VOP identifier that is unique in the component.
- marker_bit :** A 1-bit field that has a value of 1.
- first_pts :** A 33-bit numeric value that indicates the time when the first presentation unit of the I-frame/I-VOP still picture identified by this IIT is displayed. This value is specified in N units; N is the result of dividing the STC frequency by 300, or 90 kHz.
- last_pts :** A 33-bit numeric value that indicates the time when the last presentation unit of the I-frame/I-VOP still image identified by this IIT is displayed. This value is specified in N units; N is the result of dividing the STC frequency by 300, or 90 kHz.
- stream_id :** An 8-bit numeric value that indicates stream_id (0xe0 to 0xef) of the video PES to which the corresponding I-frame/I-VOP still picture belongs. A value of 0x00 means that no stream_id is specified.
- component_tag :** An 8-bit numeric value that indicates component_tag of the video PES to which the corresponding I-frame/I-VOP still picture belongs.

8.2.2 Transmission of JPEG still picture

JPEG still pictures must be transmitted through a data carousel with the stream type value of 0x0B or 0x0D.

8.2.3 Coding scheme and transmission of PNG bitmap

For the PNG bitmap data that is displayed only under the control of the CLUT data specified separately, the palette data within the PNG data can be abbreviated.

PNG bitmap graphic must be transmitted through data carousel with the stream type value of 0X0B or 0x0D.

8.2.4 Coding scheme and transmission of MNG animation

For the PNG bitmap graphic data in the MNG animation format that is displayed only under the control of CLUT data specified separately from this standard, the palette data within the PNG data can be omitted. MNG bitmap animation graphic must be transmitted through data carousel with the stream type value of 0X0B or 0x0D.

8.3 Audio Coding Scheme and Transmission

The audio coding scheme and transmission method defined in this section are applied to audio data that is referenced by the object element.

8.3.1 Transmission of MPEG-2 audio

8.3.1.1 Transmission in audio PES

To transmit MPEG-2 AAC audio through an audio PES in MPEG2-TS, it must be transmitted as a stream with the stream type value of 0x0F. To transmit MPEG-2 BC audio through an audio PES in MPEG2-TS as necessity, it must be transmitted as a stream with the stream type value of 0x03⁴ or 0x04. When streamstatus=stop is applied to an audio PES stream which is referenced by the object element, the audio data will be muted instead of being stopped.

8.3.1.2 Transmission in data carousel

To transmit MPEG-2 audio data is transmitted through a data carousel (stream type value of 0x0B or 0x0D), one of the following methods must be used.

- 1) An audio ES must be transmitted as a file.
- 2) MPEG-2 audio data is multiplexed into a time-stamped TS format file defined in Section 8.1.4.

8.3.2 Transmission of MPEG-4 audio

8.3.2.1 Transmission in audio PES

To transmit MPEG-4 Audio data is transmitted in an audio PES in MPEG2-TS, it must be transmitted as a stream with the stream type value of 0x11.

8.3.2.2 Transmission in data carousel

To transmit MPEG-4 Audio data is transmitted through a data carousel (stream type value of 0x0B or 0x0D), it must be multiplexed into a time-stamped TS format file defined in Section 8.1.4.

⁴ In the case where the stream conforms to the range of ISO/IEC 11172 specification.

8.3.3 Transmission of PCM (AIFF-C) audio

AIFF-C PCM audio must be transmitted through a data carousel with the stream type value of 0x0B or 0x0D.

8.3.4 Transmission of Additional Sound

Additional Sound must be transmitted through data carousel with the stream type value of 0x0B or 0x0D.

8.4 Character Coding and Transmission

The character coding scheme and transmission method that are defined in this section are applied to external text files that are referenced by the object element.

8.4.1 Transmission of EUC-JP text

A text file encoded in EUC-JP must be transmitted through a data carousel with the stream type value of 0x0B or 0x0D.

8.4.2 Transmission of UCS/UTF-16 text

A text file encoded in UCS/UTF-16 must be transmitted through a data carousel with the stream type value of 0x0B or 0x0D.

8.4.3 Transmission of Shift-JIS text

A text file encoded in Shift-JIS must be transmitted through a data carousel with the stream type value of 0x0B or 0x0D.

8.4.4 Transmission of 8-bit character code text

A text file containing control codes encoded in 8-bit character codes must be transmitted through a data carousel with the stream type value of 0x0B or 0x0D.

8.5 Graphic Description Command Coding Scheme and Transmission

The coding scheme and transmission method that are defined in this section are applied to external text files that are referenced by the object element.

8.5.1 Transmission of Geometric graphic data

Graphic data encoded with Geometric format must be transmitted through a data carousel with the stream type value of 0x0B or 0x0D.

8.6 External Font Coding Scheme and Transmission

The external font coding scheme and transmission method that are defined in this section are applied to external character data that is referenced by the loadDRCS() function.

8.6.1 Transmission of DRCS

External font data encoded in DRCS must be transmitted through a data carousel with the stream type value of 0x0B or 0x0D.

8.7 Transmission of Two-dimensional Table Data

Two-dimensional table coding scheme and transmission method that are defined in this section are applied to two-dimensional table data used by extended objects that are defined in Section 7.5.

8.7.1 Transmission of table data handled by a CSVTable object

Table data handled by a CSVTable object is transmitted through a data carousel with the stream type value of 0x0B or 0x0D.

8.7.2 Transmission of table data handled by a BinaryTable object

Table data handled by a BinaryTable object is transmitted through a data carousel with the stream type value of 0x0B or 0x0D.

8.8 Transmission of External XML Document

An External XML document used for an XML document object, as defined in Section 7.5, must be transmitted through a data carousel with the stream type value of 0x0B or 0x0D.

8.9 Transmission of Proprietary Data

To transmit a media type (content-type) other than the types defined in Table 9-6 in Chapter 9 of a file in a data carousel, it must be transmitted through a data carousel with the stream type value of 0x0B or 0x0D.

8.9.1 Transmission in ES with B-XML/BML Data Coding Identification

Any single file to be transmitted must be associated with a single module. This module requires the DataEncoding descriptor (See Chapter 6 in Volume 3) that contains a data encoding identification. The media type identifying the concerned proprietary data must be contained in the Type descriptor.

8.9.2 Transmission in Independent ES

To transmit proprietary data referenced from a B-XML/BML content in an independent ES, a data encoding identification identifying the concerned proprietary data must be contained in PMT. The B-XML/BML content uses component_ref in the data content descriptor in EIT to reference the ES.

Chapter 9 Content Transmission and Namespace

9.1 Transmission of Content

9.1.1 Transmission in data carousel

BXML/BML documents, monomedia data and other resources referenced by BXML/BML documents must be transmitted using a Data Carousel that is defined in ARIB STD-B24 Volume 3.

Note: However, the PSI/SI descriptor data structure defined in Section 9.3 assumes future adoptions of other transmission methods.

9.1.2 Resource-to-module mapping

This section defines the following two methods to map a resource to a module transmitted through a data carousel.

- (1) Direct mapping from one resource to one module
- (2) Storing one or more resources into a module in an HTTP/1.1 entity format defined in IETF RFC2068

However, to transmit a proprietary media type other than the types defined in Table 9-6 of a file in a data carousel, a data encoding identification of the concerned proprietary data is described with the DataEncoding descriptor. The DataEncoding descriptor is described in an MIB area in DII using the data encoding specification defined in Volume 3, or in a stored resource list using the data encoding specification defined in Section 9.1.2.4.

9.1.2.1 Module configuration to store resources in entity format

To contain a resource in an HTTP/1.1 entity format within a module, the entity consists of an entity body that contains the resource and an entity header that contains meta information. To package a resource(s) in a module, a multipart format (Content-Type:multipart/mixed) is used.

9.1.2.2 Syntax of the module data with entity format

The data must be contained in a module with the following syntax. This notation conforms to the definition in IETF RFC2068 “2. Notational Conventions and Generic Grammar.”

```
module      =  *entity-header
              CRLF
              [entity-body]
```

The definitions of entity-header and entity-body conform to IETF RFC2068, “7.1 Entity Header Fields” and “7.2 Entity Body.” To store two or more resources in a multipart format, entity-body is described as below, as defined in IETF RFC1945, “4.2.1 Multipart Types”.

```
entity-body  =  discard-text 1*encapsulation
               close-delimiter discard-text

discard-text  =  *(*text CRLF)

encapsulation =  delimiter body-part CRLF

delimiter    =  “--“ boundary CRLF

body-part    =  *entity-header
```

CRLF

[entity-body]

close-delimiter = "--" boundary "--" CRLF

In the above entity format, each entity-header of body-part must contain Content-Type: and Content-Location: to identify the type and the name of each resource.

9.1.2.3 Resource list for multipart format modules

This section defines a resource list that contains information regarding the other resources than the resource list in a module (e.g. name, offset of the start byte in package, length,). This resource list is designed to help a receiver to decode resources stored in the module in multipart format. The media type (Content-Type) of a resource list must be application/X-arib-resourceList.

Table 9-1 defines the data structure of a resource list.

Table 9-1 Encoding of Resource List

Syntax	Number of bits	Mnemonic
<pre>X-arib-resourceList { num_of_resources for (i=0; i< num_of_resources; i++) { resourceInfo() } }</pre>	8	uimsbf

Semantics of X-arib-resourceList ()

num_of_resources (Number of Resources):

This 8-bit field indicates the number of the resources in the same module.
Note that the number does not include the resource list itself.

resourceInfo() (Resource Information):

A data structure for storing information about a resource in the module, as defined in Table 9-2.

- Syntax and Semantics of Resource Information

Table 9-2 defines the data structure of resource information.

Table 9-2 Resource Information (resourceInfo())

Syntax	Number of bits	Mnemonic
<pre>ResourceInfo(){ resourceInfoLength resourceOffset header_length resourceLength resourceTypeValue() reserved_future_use resourceNameLength for (j=0; j< resourceNameLength; j++) { text_char } for (j=0 ; j< N; j++) { reserved } }</pre>	<p>8</p> <p>32</p> <p>16</p> <p>32</p> <p>16</p> <p>1</p> <p>7</p> <p>8</p> <p>8</p>	<p>uimsbf</p> <p>uimsbf</p> <p>uimsbf</p> <p>uimsbf</p> <p>bslbf</p> <p>bslbf</p> <p>uimsbf</p> <p>uimsbf</p> <p>uimsbf</p>

Semantics of Resource Information

resourceInfoLength (Resource Information Length):

This 8-bit field indicates the byte length of the area from the beginning of the immediately following field to the end of the resource information.

resourceOffset (Resource Offset):

This 32-bit field indicates the offset in bytes of the beginning of body-part of the resource from the beginning of the module.

header_length:

This 16-bit field indicates the byte length of the header area in body-part of the resource specified by this resource information. This length does not include the length of CRLF (two bytes) which is inserted as a delimiter before entity-body.

resourceLength:

This 32-bit field indicates the length of the resource specified by this resource information. This value must be equivalent to the value of Content-Length field in the header area of body-part.

resourceTypeValue() (Resource Media Type):

This is a 16-bit data structure. It indicates the media type of the resource specified by the resource information. The detailed data structure is defined in Table 9-3.

resourceNameLength (Resource Name Length):

This 7-bit field indicates the length in bytes of the following resource name. If the resource name has less than 127 bytes length, padding (code 0x00) may be occurred.

text_char (Resource Name):

This 8-bit field contains a string representing a resource name. The string is up to 127 bytes. The resource name specified with this field must be equivalent to the value stored in Content-Location in the header area of body-part.

- Syntax and Semantics of Resource Media Type

Table 9-3 Resource Media Type

Syntax	Number of bits	Mnemonic
ResourceTypeValue() { typeValue subtypeValue }	4 12	bslbf bslbf

Semantics of Resource Media Type

typeValue (File Type):

This 4-bit field indicates a file type. The file type must correspond to the value for Type component in Content-Type field of the resource header. Table 9-4 lists the available values of this field and the corresponding Type values.

Table 9-4 File Type Values

File Type Value	Corresponding Type
0x0	multipart
0x1	text
0x2	image
0x3	audio

File Type Value	Corresponding Type
0x4	video
0x5	application
0x6	message
0x7	model
0x8 ~ 0xF	reserved

subtypeValue (Format Type):

This 12-bit field indicates a subtype. Table 9-5 lists the available subtype values.

Table 9-5 Subtype Values

SubtypeValue	Subtype
0x000 ~ 0x3FF	Subtype defined by IANA
0x400 ~ 0x7FF	Subtype defined by ARIB (X-arib-****)
0x800 ~ 0xFFF	A range of values that can be defined by the broadcaster

Table 9-6 lists the available media types in a resource header. Each media type is followed by its corresponding file type field value and format type field value in the resource media type data structure.

Table 9-6 Media type (Content-Type) and Corresponding File Type and Format Type

Media Type	File Type	Format Type
text/xml; charset="euc-jp"	0x1	0x011
text/xml; charset="UTF-16"	0x1	0x012
text/xml; charset="Shift_JIS"	0x1	0x013
text/css	0x1	0x020
text/xsl; charset="euc-jp"	0x1	0x031
text/xsl; charset="UTF-16"	0x1	0x032
text/xsl; charset="Shift_JIS"	0x1	0x033
text/html; charset="euc-jp"	0x1	0x041
text/html; charset="UTF-16"	0x1	0x042
text/html; charset="Shift_JIS"	0x1	0x043
text/X-arib-bml; charset="euc-jp"	0x1	0x401
text/X-arib-bml; charset="UTF-16"	0x1	0x402
text/X-arib-bml; charset="Shift_JIS"	0x1	0x403
text/plain; charset="euc-jp"	0x1	0x001
text/plain; charset="UTF-16"	0x1	0x002
text/plain; charset="Shift_JIS"	0x1	0x003
text/X-arib-jis8text	0x1	0x411
text/X-arib-ecmascript; charset="euc-jp"	0x1	0x421
text/X-arib-ecmascript; charset="UTF-16"	0x1	0x422
text/X-arib-ecmascript; charset="Shift_JIS"	0x1	0x423
application/X-arib-meta+xml; charset="UTF-8"	0x1	0x424
application/X-arib-meta+xml; charset="UTF-16"	0x1	0x425
image/jpeg	0x2	0x000
image/png	0x2	0x001
image/gif	0x2	0x002
image/X-arib-png	0x2	0x401
image/X-arib-mng	0x2	0x402
image/X-arib-mpeg2-I	0x2	0x403

Media Type	File Type	Format Type
image/X-arib-mpeg4-I-simple	0x2	0x411
image/X-arib-mpeg4-I-core	0x2	0x412
image/X-arib-H264-I-baseline	0x2	0x421
image/X-arib-H264-I-main	0x2	0x422
audio/X-arib-mpeg2-aac	0x3	0x411
audio/X-arib-mpeg2-bc	0x3	0x412
audio/X-arib-aiff	0x3	0x401
audio/X-arib-additional	0x3	0x421
video/X-arib-mpeg1	0x4	0x401
application/xhtml+xml; charset="euc-jp"	0x5	0x044
application/xhtml+xml; charset="Shift_JIS"	0x5	0x045
application/X-arib-stream-text; charset="euc-jp"	0x5	0x401
application/X-arib-stream-text; charset="UTF-16"	0x5	0x402
application/X-arib-stream-text; charset="Shift_JIS"	0x5	0x403
application/X-arib-stream-jis8text	0x5	0x411
application/X-arib-stream-png	0x5	0x421
application/X-arib-stream-jpeg	0x5	0x422
application/X-arib-stream-mpeg2-I	0x5	0x423
application/X-arib-mpeg2-tts	0x5	0x424
application/X-arib-stream-mpeg4-I-simple	0x5	0x425
application/X-arib-stream-mpeg4-I-core	0x5	0x426
application/X-arib-stream-H264-I-baseline	0x5	0x427
application/X-arib-stream-H264-I-main	0x5	0x428
application/X-arib-bmlclut	0x5	0x431
application/X-arib-btable	0x5	0x441
application/X-arib-drcs	0x5	0x451
application/X-arib-PDI	0x5	0x461
application/X-arib-resourceList	0x5	0x471
application/X-arib-storedResourceList	0x5	0x472
application/X-arib-rootcertificate	0x5	0x491

9.1.2.4 Stored resource list for multipart format modules assuming stored services

This section defines a stored resource list that contains information regarding the other resources than the stored resource list in a module (e.g. name, offset of the start byte in package, length,) as well as a directory structure to be formed on a storage media that stores the concerned resources. This stored resource list is designed to support a stored data services or other data storing services and to help a module to obtain resources from a storage media that is on a receiver and contains modules. The media type (Content-Type) of a stored resource list must be application/X-arib-storedResourceList.

Table 9-1 defines the data structure of a stored resource list.

Table 9-7 Encoding of Stored Resource List (X- arib-storedResourceList)

Syntax	Number of bits	Mnemonic
X-arib-storedResourceList {		
storedResourceListLength	32	uimsbf
additionalDirectoryInfoLength	16	uimsbf
for (j=0; j<N; j++) {		
additionalDirectoryInfo	8	uimsbf
}		
num_of_files	16	uimsbf

for (i=0; i< num_of_files; i++) { directoryFlag reserved if (directoryFlag == "1") { storedDirectoryInfo() } else { storedFileInfo() } }	1 7	bslbf bslbf
---	--------	----------------

Semantics of X-arib-storedResourceList ()

storedResourceListLength (stored resource list length):

This 32-bit field indicates the number of bytes in the area from the beginning of the immediately following field to the end of the stored resource list.

additionalDirectoryInfoLength (additional directory information length):

This 16-bit field indicates the number of bytes of the following additional directory information.

additionalDirectoryInfo (additional directory information):

This 8-bit field contains descriptors and values, as required, described in Table 9-10, which is part of the data structure of the descriptors defined Section 6.2.3, Volume 3.

num_of_files (number of files):

The number of directories and files located under the concerned directory.

DirectoryFlag (directory flag):

This 1-bit field indicates whether an element located immediately under the concerned directory is a file or a directory.

Value	Semantics
0	The element is a file (resource).
1	The element is a directory.

storedDirectoryInfo() (stored file information):

This field contains information about the concerned stored file, as defined in Table 9-9.

Table 9-8 Encoding of storedDirectoryInfo()

Syntax	Number of bits	Mnemonic
storedDirectoryInfo() { storedDirectoryInfoLength reserved directoryNameLength for (j=0; j< directoryNameLength; j++) { text_char } additionalDirectoryInfoLength for (j=0; j<N; j++) { additionalDirectoryInfo } num_of_files for (i=0; i< num_of_files; i++) { directoryFlag reserved	32 1 7 8 16 8 16 1 7	uimsbf bslbf uimsbf uimsbf uimsbf uimsbf uimsbf bslbf bslbf

<pre> if (directoryFlag == "1") { storedDirectoryInfo() } else { storedFileInfo() } } </pre>		
--	--	--

Semantics of storedDirectoryInfo():

storedDirectoryInfoLength (Stored Directory Information Length):

This 32-bit field indicates the byte length of the area from the beginning of the immediately following field to the end of the stored directory resource information.

directoryNameLength (directory name length):

This 7-bit field indicates the length of the following directory name.

text_char (directory name):

This 8-bit field contains a string representing a directory name. The string is up to 127 bytes.

additionalDirectoryInfoLength (additional directory information length):

This 16-bit field indicates the number of bytes of the following additional directory information.

additionalDirectoryInfo (additional directory information): This 8-bit field contains descriptors and values, as required, described in Table 9-10, which is part of the data structure of the descriptors defined Section 6.2.3, Volume 3.

num_of_files (number of files):

The number of directories and files located under the concerned directory.

directoryFlag (directory flag):

This 1-bit field indicates whether an element located immediately under the concerned directory is a file or a directory.

Value	Semantics
0	The element is a file (resource).
1	The element is a directory.

storedDirectoryInfo() (stored directory information):

This field contains information about a directory located immediately under the concerned directory. The stored directory information is presented recursively based on the directory structure.

stored FileInfo() (stored file information):

This field contains information about the concerned stored file, as defined in Table 9-9.

Table 9-9 Encoding of storedFileInfo()

Syntax	Number of bits	Mnemonic
storedFileInfo() {		
fileInfoLength	16	uimsbf
resourceOffset	32	uimsbf
header_length	16	uimsbf
resourceLength	32	uimsbf
resourceTypeValue()	16	bslbf
reserved_future_use	1	bslbf
fileNameLength	7	uimsbf

for (j=0; j< fileNameLength; j++) { text_char }	8	uimsbf
additionalFileInfoLength	16	uimsbf
for (j=0 ; j< N; j++) { additionalFileInfo }	8	uimsbf
}		

Semantics of storedFileInfo ():

fileInfoLength (File Information Length):

This 16-bit field indicates the byte length of the area from the beginning of the immediately following field to the end of the resource information.

resourceOffset (Resource Offset):

This 32-bit field indicates the offset in bytes of the beginning of body-part of the resource from the beginning of the module.

header_length: This 16-bit field indicates the byte length of the header area in body-part of the resource specified by this resource information. This length does not include the length of CRLF (two bytes) which is inserted as a delimiter before entity-body.

resourceLength: This 32-bit field indicates the length of the resource specified by this resource information. This value must be equivalent to the value of Content-Length field in the header area of body-part.

resourceTypeValue() (Resource Media Type):

This is a 16-bit data structure. It indicates the media type of the resource specified by the resource information. The detailed data structure is defined in Table 9-3.

fileNameLength (File Name Length):

This 7-bit field indicates the length in bytes of the following file name.

text_char (File Name):

This 8-bit field contains a string representing a file name. The string is up to 127 bytes. The file name specified with this field must be equivalent to the resource name stored in Content-Location in the header area of body-part. When resource names are separated with "/", the file name is equivalent to a string preceded by the last "/".


additionalFileInfoLength (additional file information length):

This 16-bit field indicates the number of bytes of the following private file data area.

additionalFileInfo (additional file information):

This 8-bit field contains descriptors and values, as required, described in Table 9-10, which is part of the data structure of the descriptors defined in Section 6.2.3, Volume 3.

**Table 9-10 Available Values and Descriptors to additionalDirectoryInfo
/additionalFileInfo**

Tag value	Descriptor	Semantics	additionalDirectoryInfo	additionalFileInfo
0x01	Reserved		-	-
0x02	Reserved		-	-
0x03	Info descriptor	Character type of a file/directory	O	O
0x04	Reserved		-	-
0x05	Reserved		-	-
0x06	Reserved		-	-
0x07	Reserved		-	-
0x08 ~ 0x7F	Reserved for future use		-	-
0x80 ~ 0xBF	The available tag values to a descriptor defined by a broadcaster		-	-
0xC0	Expire descriptor	Expiration date of a directory/file	O	O
0xC1	ActivationTime descriptor	Activation time of a directory/file	O	O
0xC2	Compression Type descriptor	Used compression algorithm, if any	-	O
0xC3	Control descriptor	Information required to control/interpret a directory/file	O	O
0xC4	ProviderPrivate descriptor	Used to add proprietary information by a network provider/broadcaster	O	O
0xC5	Reserved		-	-
0xC6	Reserved		-	-
0xC6	Reserved		-	-
0xC7	Title descriptor	Used to indicate a title to describe a module/directory/file in a list to be presented to end users	O	O
0xC8	DataEncoding descriptor	Used to identify the data coding specification for proprietary data in a resource	-	O
0xC9	Time-stamped TS descriptor	Used to add information for transmission of MPEG video/audio in a time-stamped TS format defined in Section 8.1.4.3, Volume 2, in a data carousel	-	O
0xCA	Root certificate descriptor	Used to identify a root certificate, which is contained in a module and is applied to an  raction channel telecommunication process[MEI11]	-	-
0xCB	Encrypt descriptor	See Part 2, ARIB STD-B25		
0xCC	ACG descriptor	See Part 2, ARIB STD-B25		
0xCD ~ 0xED	Reserved for future use			

Tag value	Descriptor	Semantics	additionalD irectoryInfo	additional FileInfo
0xEE	MetadataFragmen t descriptor	Used to identify the ID and version of metadata contained in a module	-	O
0xEF	TransportLocation descriptor	See Table E-5	-	O
0xF0 ~ 0xFF	Reserved for descriptor tags identifying a data coding specification		O	O

9.1.3 Transmission of event messages

Messaging from the broadcasting station to a BML/B-XML document being deployed on the receiver must be transmitted using the Event Message transmission method defined in ARIB STD-B24 Volume 3.

9.1.3.1 Mapping between event messages and events defined in BML

- Component through That Event Messages are Transmitted

When an event message is received, the URI string of the component in which the event message section is transmitted must be mapped to esRef attribute of BMLBeventEvent interface that is defined as an event interface. On the other hand, when the BML content waits for an event message with "EventMessageFired" for type attribute of beitem element specified, the URI string of the component is to be specified within es_ref attribute according to the notation defined in Section 9.2.1.

- Relationship between version_number in DSM-CC Section and Generic Event Message Descriptor

The DSM-CC section that transmits an event message is designed to carry two or more generic event message descriptors, if required, that are assumed to be updated/modified separately. In this case, when any change or modification is applied to at least one descriptor, the section version is incremented by 1. In a single DSM-CC section, the event messages are fired in the same order in which the corresponding generic event message descriptors are described in the section.

- Mapping between Generic Event Message Descriptors and Events

Table 9-7 shows mapping between the event message descriptor fields and their corresponding events defined in BML.

Table 9-11 Mapping between Generic Event Message Descriptors and Events

Generic Event Message Descriptor Field	Mapping to BML
Indicating time according to time_mode	If time_mode=0, the event occurs immediately after the event message is received. For other cases, the event occurs at a time specified by the descriptor. If time_mode or time_value is omitted, the timer is not fired.
event_msg_group_id	Mapped to eventMsgGroupId attribute of BMLBeventEvent interface that is defined as an event interface. If the procedure in the case when the event message is fired is designated using the description of the beitem element, it is mapped to the event_msg_group_id attribute of the beitem element.

event_msg_id	event_msg_id is divided into two parts. The upper eight bits are used for identifying an event message. The lower eight bits are used for identifying the version number of the event message. When an event message is invoked, above two identifiers must be mapped to MessageId and MessageVersion attributes of BMLBeventEvent interface defined as an event interface. If the procedure which invoked by an event message is specified with beitem event, message_id and message_version attributes of beitem element are to be used.
private_data_byte	Mapped to privateData attribute of BMLBeventEvent interface defined as an event interface.

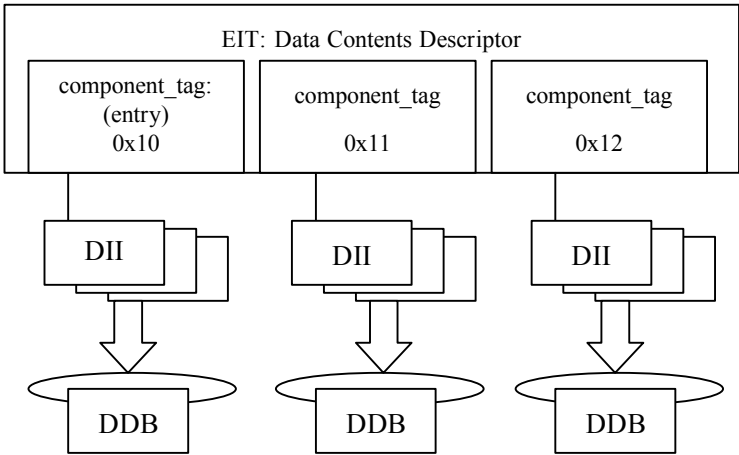
9.2 Namespace

9.2.1 Component structure and resource namespace in event

9.2.1.1 Component structure

A carousel is transmitted through a component ES. Therefore, a carousel is identified with the transmitting component tag.

Example: Figure 9-1 shows a sample structure of a carousel for a BML content that is transmitted in an event.



Note: Values of component_tag are examples for description purpose.

Figure 9-1 Sample Carousel Containing BML Content

9.2.1.2 Namespace of resource in network

The namespace for a resource transmitted through a data carousel is defined as follows.

- Names used for identifying Module

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

arib-dc://<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. 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. However, no character (string) identifying the hexadecimal notation, including a prefix “0x” and a postfix “h” is used. Instead, the numeric character, 0 (zero), is padded from the beginning of the ID field, as required to make an identification a specified fixed length.
For example, if component_tag is 0x04, specify as follows.
 <component_tag> : 04
- A moduleId used for <moduleName> is a hexadecimal character string. However, no character (string) identifying the hexadecimal notation, including a prefix “0x” and a postfix “h” is used. Instead, the numeric character, 0 (zero), is padded from the beginning of the ID field, as required to make a moduleId the specified fixed length, 4-bit. For example, if module number is 0x0001, specify as follows.
 <moduleName> : 0001

- Identification of Resource Directly Mapped to Module

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

- Identification of a Resource Stored in a Module in Entity Format

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

arib-dc://<original_network_id>.<transport_stream_id>.<service_id>[;<content_id>]
[.<event_id>]/<component_tag>/<moduleName>/<resourceName>

In this description, <resourceName> is equivalent to the character string specified in Content-Location: of the resource entity header. <resourceName> is case insensitive. For example, “abc” and “ABc” indicate the same resource. The available characters to <resourceName> are listed below.

resourceName	=	startChar *echar
echar	=	startChar "-" "." "!" "~" "" "(" ")" ";" "/" "@" "=" "+" "\$" ","
startChar	=	lowalpha upalpha digit “_”
lowalpha	=	"a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"
upalpha	=	"A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z"
digit	=	"0" "1" "2" "3" "4" "5" "6" "7" "8" "9"

9.2.1.3 Abbreviated names

- Abbreviations for reference within event

- (1) When a name is specified in the form `/<component_tag>/<moduleName>[/<resourceName>]`, it is interpreted as:

`arib-dc://....[<event_id>]/<component_tag>/<moduleName>[/<resourceName>]`

- (2) A tilde “~” may be used to represent a component tag of an ES that transmits a referencing BML document. For example, when a BML document reference a resource that is transmitted in the same ES in which the BML document is transmitted, the component tag of the ES may be specified as: `~/<moduleName>[/<resourceName>]`.

- Abbreviations for reference from BML document mapped to module

To reference a module from a BML document that is mapped to a component, a name of the module may be described relatively to the component that transmits the module.

- (1) When a name of a module is described as `<moduleName>` in a BML document that is mapped the module, it is interpreted as:

`arib-dc://....[<event_id>]/<component_tag>/<moduleName>/`

- (2) When a name of a module is described as `<moduleName>/<resourceName>` in a BML document that is mapped the module, it is interpreted as:

`arib-dc://....[<event_id>]/<component_tag>/<moduleName>[/<resourceName>]`

- Abbreviations for reference from BML document in module in entity format

To reference a resource from a BML document that is contained in a module in an entity format, a name of the resource may be described relatively to the module. In this case, a name of a resource described as `<resourceName>` in the BML document is interpreted as:

`arib-dc://....[<event_id>]/<component_tag>/<moduleName>/<resourceName>`

9.2.2 Startup BML/B-XML document

A BML/B-XML document that is launched before the other documents in a data carousel in a component ES must contain a module to which the 0x0000 module number is assigned. When a module with 0X0000 module number is in an entity format, a BML/B-XML document whose `<resourceName>` is “startup.bml” is launched first.

9.2.3 Reference of AV streams and subtitle component

To reference an AV stream transmitted in MPEG-2 TS or a component transmitting subtitle, the referenced stream or component is uniquely identified in the network with the following naming convention.

`arib://<original_network_id>.<transport_stream_id>.<service_id>[;<content_id>]
[.<event_id>]/<component_tag>[;<channel_id>]`

The `<component_tag>` field has a of component tag (`component_tag`) that is defined in ARIB STD-B10 in the hexadecimal notation. However, no character (string) identifying the hexadecimal notation, including a prefix “0x” and a postfix “h” is used. Instead, the numeric character, 0 (zero), is padded from the beginning of the field, as required to make a description the specified fixed length, 2-bit. When `<component_tag>` is set to a special value, “-1,” it indicates an AV stream that is selected by EPG and others.

The `<channel_id>` filed is applicable only to a dual monaural audio stream. The value 1 indicates the first channel, The value 2 indicates the second channel. The value 3 means that the first and second channels are used for simultaneous playback.

9.2.3.1 Abbreviated AV stream names

When an AV stream is specified in the following format:

/<component_tag>[;<channel_id>]

It is interpreted as:

arib://....[<event_id>]/<component_tag>[;<channel_id>]

Further, when an AV stream in a file format is transmitted using the data carousel scheme, the shortened form is specified in the following format:

/<component_tag>/<moduleName>[/<resourceName>]

It is interpreted as:

arib://....[<event_id>]/<component_tag>/<moduleName>[/<resourceName>]

9.2.4 Identification of MPEG-I frames transmitted through a still picture carousel

Any MPEG-I frame transmitted through a still picture carousel, as defined in Chapter 8, is uniquely identified by the following naming convention.

*arib-ic://<original_network_id>.<transport_stream_id>.<service_id>[;<content_id>]
[.<event_id>]/<component_tag>/<I-frame_ID>*

The *<component_tag>* field has a string in the hexadecimal notation. The *<component_tag>* field represents a component tag (*component_tag*) of a component transmitting an IIT component defined in Section 8.4.3. The *<I-frame_ID>* field is an I-frame identifier that is unique in the component. It must be described in the hexadecimal notation. However, no character (string) identifying the hexadecimal notation, including a prefix "0x" and a postfix "h" is used. Instead, the numeric character, 0 (zero), is padded from the beginning of the field, as required to make a description the specified fixed length, 4-bit.

9.2.4.1 Abbreviated still picture carousel names

When an MPEG I-frame that is transmitted through a still picture carousel is specified with an abbreviation of the following format,

"/<component_tag>/<I-frame_ID>"

It is interpreted as follows:

"arib-ic://....[<event_id>]/<component_tag>/<I-frame_ID>"

9.2.5 Service identification

The following character string is used to reference a service.

arib://<original_network_id>.<transport_stream_id>.<service_id>

9.2.5.1 Identification of currently selected broadcasting service on receiver

When the fields *<original_network_id>*, *<transport_stream_id>*, and *<service_id>* contain a special value, -1, it is interpreted to mean that a broadcasting service currently selected on a receiver is specified.

When an object element whose type attribute is video/X-arib-mpeg2 or audio/X-arib-mpeg2-aac has the following string as the data attribute, it is interpreted to mean that a default video stream of a broadcasting service currently selected on a receiver is specified:

arib://-1.-1.-1/-1

9.2.6 Event identification

The following character string is used to reference an event.

arib://<original_network_id>.<transport_stream_id>.<service_id>.<event_id>

9.2.7 Identification of stored contents

The following character string is used to reference a group of stored contents.

arib://<original_network_id>.<transport_stream_id>.<service_id>;<content_id>

9.2.8 Identification of ERT node of Local Event Indexes

The following character string is used to reference a node in ERT.

arib-node://<original_network_id>.<transport_stream_id>[.<service_id>[.<event_id>]]
/<information_provider_id>/<event_relation_id>/<node_id>

For more information about a local event index and an event group index, see ARIB STD-B10 Version 1.2, Part 3.

9.2.8.1 Abbreviated identifications of ERT nodes

When an ERT node is specified using an abbreviation with the following format:

/<information_provider_id>/<event_relation_id>/<node_id>

It is interpreted as follows:

arib-node://....[<event_id>]/<information_provider_id>/<event_relation_id>/<node_id>

9.2.8.2 Grouping information of event group index

To identify grouping information for an event group index, <service_id> and <event_id> are not specified.

9.2.9 Names of sound built in the receiver

The following character string is used to identify additional sound built in the receiver:

romsound://<sound_id>

where <sound_id> identifies the type of built-in sound that is defined in an operational standard regulation defined separately from this standard.

9.2.10 Namespace of persistent memory

Any name of a file in a persistent memory device is described in the following format.

- For NVRAM:

nvrnm://<filename>

To read/write the data in NVRAM, non-volatile memory functions defined in Section 7.6.5.1, as required, are used. The functions for controlling access-controlled areas defined in Section 7.6.5.2 are not applicable to accessing any file that is identified with nvrnm:.

- For NVRAM with access control information:

nvrnm://<filename>

To read/write the data in NVRAM, and set the access control information for NVRAM, functions for controlling access-controlled areas, defined in Section 7.6.5.2, as required, are used. The detail of the access control information is defined in an operational standard regulation defined separately from this standard. The non-volatile memory functions defined in Section 7.6.5.1 are not applicable to accessing any file that is identified with nvrnm:.

- For bookmark area:

nvrnm://bookmark/<block_number>

To read/write the data in a bookmark area, functions for controlling bookmark areas defined in Section 7.6.13, as required, are used.

9.2.11 Identification of component ES transmitting event message

The following character string is used to reference the component ES which transmits the NPT reference descriptor and generic event message descriptor defined in ARIB STD-B24 Volume 3 Chapter 7, as is the same format of namespace defined for the identification of resource and module in the Section 9.2.1. The string ending with <component_tag> is used.

arib-dc://<original_network_id>.<transport_stream_id>.<service_id>
[;<content_id>][.<event_id>]/<component_tag>

9.2.11.1 Abbreviated name of component ES transmitting event message

When a component ES transmitting the NPT reference descriptor and generic event message descriptor is specified with

/<component_tag>

, it is interpreted as the following:

arib-dc://....[<event_id>]/<component_tag>

And, when the component ES transmitting the NPT reference descriptor and generic event message descriptor is the default ES in the content, it can be specified as follows:

~

9.2.12 Identification of series

The following character string is used to reference the series specified by the series descriptor.

arib-series://<series_scope_ref>/<series_id>

<series_id> is specified with the value of series_id in the series descriptor. It must be described in the hexadecimal notation. However, no character (string) identifying the hexadecimal notation, including a prefix "0x" and a postfix "h" is used. Instead, the numeric character, 0 (zero), is padded from the beginning of the ID field, as required to make a moduleID the specified fixed length, 4-bit

<series_scope_ref> is the character string which specifies the range where service_id can be used uniquely and is specified separately in an operational standard regulation.

9.2.13 Namespace used for obtaining resources through an IP packet transmission line

Such resource name as the URI string starting with http: or ftp: may be used when the resource is obtained via the IP network.

9.2.14 Data storage and Name descriptor values

Figure 9-2 shows a sample mapping of the bunch of modules (see Figure 9-1) that are transmitted with multiple ESes through a broadcast network to file hierarchy stored on a storage device. All data of this data broadcasting program is placed under the content_id directory. Modules that are transmitted through a carousel are stored under the content_id directory in a subdirectory for component_tag value with the main broadcasting stream (video and audio) and Local-event Information Table.

As this mapping indicates, the modules can share the same namespace both in broadcast network and in the storage device.

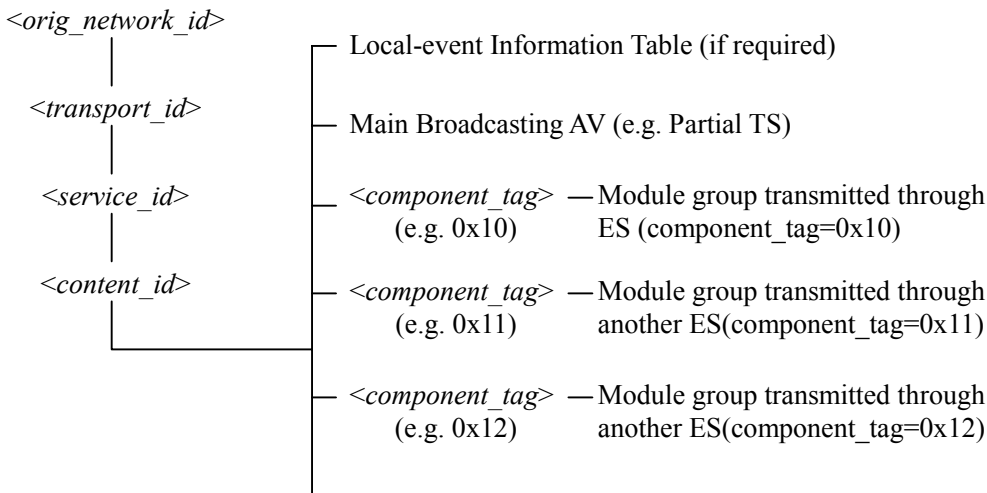


Figure 9-2 Sample Mapping of Transmission Contents onto Storage Device

9.2.15 Namespace convention for file in storage media

The namespace convention defined in Section 9.2.10 is designed to specify a name of a file for a data broadcasting service on a storage device. This section defines a namespace convention applicable to a content that is intended to be stored and is transmitted in more than one service. This section also defines a namespace convention that is independent of a service network to enable a file management function to specify a location to which a file is copied.

It is assumed that a number and a configuration/structure of storage devices vary with an individual receiver. A logical path, which is independent of a device, is required to reference a resource on a storage media. A physical location in which a resource identified with a logical path is actually contained depends on the concerned device.

9.2.15.1 Specification of logical path

To specify a logical path of a resource in a storage media, the following format is used.

arib-file:// *(<directoryName>/)<fileName>

<directoryName> :directory Name

<fileName> :file Name

The available characters to <resourceName>/<fileName> are listed below.

directory Name/ file Name = startChar*echar

echar = startChar | "-" | "."

startChar = lowalpha | upalpha | digit | "_"

lowalpha = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" |
"m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" |
"y" | "z"

upalpha = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" |
"L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" |
"W" | "X" | "Y" | "Z"

digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

- <resourceName>/<fileName> is case-insensitive.
- The character "/" is used to identify a level of a directory hierarchy.
- Any directory/file name is a string whose minimum length is 1-character and maximum length is 16-character.

9.2.15.2 Abbreviated Names

The character "." represents a directory to which a currently presented BML document belongs. An abbreviated name starting with a directory/file name means that this abbreviated name is a relative name to a directory to which a currently presented BML document belongs.

For example, when a BML document identified with arib-file://DirA/DirB/DirC/DirD/xx.bml contains a description as the following:

./DirE/yy.bml

or

DirE/yy.bml

It is interpreted as the following:

arib-file:// DirA/DirB/DirC/DirD/DirE/yy.bml

The character ".." represents a directory located immediately above a directory to which a currently presented BML document belongs. An abbreviated name starting with a directory/file name means that this abbreviated name is a relative name to a directory to which a currently presented BML document belongs.

For example, when a BML document identified with arib-file://DirA/DirB/DirC/DirD/xx.bml contains a description as the following:

../DirF/yy.bml

It is interpreted as the following:

arib-file://DirA/DirB/DirC/DirF/yy.bml

9.2.15.3 Namespace for stored data services

- Ensuring uniqueness of content

To store a content in a stored data service in which DII contains the StoreRoot descriptor and the SubDirectory descriptor, the StoreRoot descriptor must contain a special two-tier directory, “<rootName>/<subrootName>”. In this case, a file in a storage media is identified with the following format that uses a logical path defined in section 9.2.15.1.

<rootName> :Root directory
<subrootName> :Directory immediately under <rootName>
<directoryName> :Directory name
<fileName> :File name

The following is applicable to this format:

- It is assumed that rootName is used to identify a broadcaster and subrootName is used to identify a program (or a series of programs). More detailed usage of rootName and subrootName is defined separately from this standard.
- The uniqueness of a content in a storage device is ensured by the uniqueness of subrootName the root directory. More detailed usage of root directory name to ensure the uniqueness is defined separately from this standard in an operational standard regulation.
- A module in a local content is stored in a directory identified with a location that is immediately under a directory identified with the above format added with the SubDirectory descriptor.

Note: The subrootName descriptor is designed to share a content among stored data services and real-time broadcasting services by using a name described with the StoreRoot descriptor and the SubDirectory descriptor. For more information , see Informative Explanation 4.

- Mapping module in carousel to file on storage media

Any mapping a module in a data carousel to a file on a storage media complies with the following:

- To map a resource directly to a module:
 - The module is recognized as a file on a storage media.
 - A string described in the Name descriptor is used as a file name.
 - When the Name descriptor is omitted, moduleId is used as a file name.
- To store a resource in an entity format in a module:
 - The module is recognized as a directory on a storage media and any resource in the module is recognized as a file located immediately under the directory.
 - A string described in the Name descriptor is used as a directory name.
 - When the Name descriptor is omitted, moduleId is used as a directory name.
 - Each resource name is used as the corresponding file name. When a resource name contains the character “/”, a single occurrence of “/” results in a single subdirectory. For example, when a resource has a resource name "AAA/BBB", the resource is recognized as a file with a filename "BBB", located immediately under a subdirectory with a subdirectory name "AAA", that is immediately under a directory corresponding to the module.

When the Name descriptor is omitted and moduleId is used as a directory/file name, the directory/file name is a string in the hexadecimal notation. However, no character (string) identifying the hexadecimal notation, including a prefix “0x” and a postfix “h” is used. Instead, the numeric character, 0 (zero), is padded from the beginning of the field, as required to make the string the specified fixed length, 4-bit. For example, when a module number is 0x0001, the corresponding directory/file name is 0001.

For the purpose of stored data services, the description of a string used for the StoreRoot descriptor, the SubDirectory descriptor, the Name descriptor, and a resource name complies with the conventions defined in Section 9.2.15.1., except that the character "/" is not used for the Name descriptor.

- Abbreviated names

For the purpose of stored data services, the description of abbreviated names complies with the conventions in Section 9.2.15.3 and the following:

- An abbreviated name starting with the character "/" means that this abbreviated name is a relative name to a directory identified with `<rootName>/<subrootName>`.

For example, when a BML document contains a description as the following:

/DirC/yy.bml

It is interpreted as the following:

arib-file://_rootService/subA/DirC/yy.bml

9.2.16 Namespace and reference convention for time-stamped TS format AV file and stored TS file

- Referencing a time-stamped TS format AV file and the contained AV streams, and a stored TS file and the internal A/V

To reference time-stamped TS format AV files or other AV streams, that are transmitted in a data carousel, as defined in Section 8.1.4, and a stored TS file from a BML document, the following namespace convention is used:

- (A) An AV stream to be transmitted is declared with the declare attribute of the object element. A program number is assigned to the object element by using the param element.
- (B) An object element that references an individual video/audio stream uses the param element to reference the AV stream associated with the param element, as described above, and the object declared with a program number in the AV stream.

- Namespace for a time-stamped TS format file and a stored TS file

The object element described in the above (A) has the data attribute whose value is a name identifying a time-stamped TS format file or a stored TS file. For the purpose of referencing the time-stamped TS format file, the following namespace convention is used:

- A name starting with "arib-dc:" and its abbreviated name, as defined in Section 9.2.1, may be used.
- A name containing "http:", "https:", or "ftp:" and its abbreviated name, as defined in Section 9.2.13, may be used.
- A name starting with "arib-file:" and its abbreviated name, as defined in Section 9.2.15, may be used. In this case, a program number is required to be specified in the param element of a child element of the object element.
- The value of the name element used to specify a program number is "program_number".
- A string in the hexadecimal notation representing the program number is specified for the value attribute corresponding to the name attribute of "program_number" described above. However, no character (string) identifying the hexadecimal notation, including a prefix "0x" and a postfix "h" is used. Instead, the numeric character, 0 (zero), is padded from the beginning of the field, as required to make the string the specified fixed length, 4-bit.
- The program number "-1" means that the program is the first program that occurs in PAT contained in the AV file.

To name a stored TS file, a name that starts with "arib-file:", as defined in Section 9.2.15, or the corresponding abbreviated name must be used.

- Namespace for video/audio stream in a time-stamped TS format file and a stored TS file

When the object element described in the above (B) references a time-stamped TS format file or a stored TS file declared as described in above (A), the (B) object element uses the param element of a child element of the (B) object, as described below:

- The value of the name attribute used to reference the (A) object element is "stream_ref".
- The value of the value attribute corresponding to the name attribute whose value is "stream_ref", as described above, is the value of the id attribute of the (A) object.

In this case, to use this (B) object element to reference an individual video/audio stream, component_tag of a component transmitting the concerned stream in the abbreviated form ("/<component_tag>"), as defined in Section 9.2.11.1.

For example:

```
<object declare="declare"
  id="system1"
  data="/40/streamfile"
  type="application/X-arib-mpeg2-tts">
  <param name="program_number" value="0010"
    valuetype="data" />
</object>
:
:
<object id="video1" data="/01"
  type="video/X-arib-mpeg4-simple">
  <param name="stream_ref" value="system1"
    valuetype="object" />
</object>
<object id="audio1" data="/10"
  type="audio/X-arib-mpeg2-aac">
  <param name="stream_ref" value="system1"
    valuetype="object" />
</object>
```

9.2.17 Namespace for external devices

To identify an external device for exchanges of an External EML document with the external device by using an XML document object defined in Section 7.5.3 or other operations, URI described in the following format is used:

```
peripheral://<peripheral_id>
```

This format of URI depends on an individual external device and the connection between the external device and a receiver. To prepare a content, instead of an explicitly description, URI is obtained by

peripheral://CarNavigation1 using the function. The available descriptors are defined in an operational standard regulation.

For example:

peripheral://CarNavigation1

9.2.18 Namespace convention for identifying server-based content

9.2.18.1 Identification of server-based content

A CRID (Content Reference ID) is used to identify server-based content, as described in 4.1.1, ARIB STD-B38. The following character string is used to identify server-based content, as specified in 4.1.1, ARIB STD-B38.

crid://<authority>/<data>

The <data> can be used to identify a directory and a resource in server-based content. Actual usage of <data> is defined in an operational standard regulation.

9.2.18.2 Identification of video scene

A video scene in server-based content is defined in segment information, as specified in 3.2.5.3, ARIB STD-B38. This implies that segmentId, which is the identification of a segment information element, is used to identify a video scene in server-based content. The delimiter ";" is used to describe a series of segments, allowing two or more segments to be placed in a series.

arib-seg:[<authority>/]<segmentId>*(<authority>/<segmentId>)

The identification of a segment group, groupId, is used to identify a group of segments described in 3.2.5.4, ARIB STD-B38.

arib-segrp:[<authority>/]<groupId>

9.3 Data Structure of PSI/SI Descriptor That Depends on Transmission of B-XML/BML Contents

9.3.1 Identification of data coding scheme

The B-XML and BML encoding schemes share a single value as the Data Component Identifier (data_component_id). The value applicable to data_component_id is defined in a separate document.

To save only a piece of monomedia data, which is a standard component of the BML encoding scheme, into a memory card or other storage for an independent usage, by using a file management function defined in Chapter 7, the piece of monomedia data is interpreted as a single service. To identify this usage, a specific value other than the value for representing the B-XML and BML encoding schemes must be used. The value applicable to this purpose is defined in a separate document.

9.3.2 Information encoded in additional identification information area of data coding scheme descriptor

If the Data Component Identifier specifies B-XML/BML, an additional_arib_bxml_info() data listed in Table 9-8 must be encoded in the additional_data_component_info area of the Data Component Descriptor which appears in the PMT.

Table 9-12 Additional_arib_bxml_info()

Syntax	Number of bits	Mnemonic
Additional_arib_bxml_info(){		
Transmission_format	2	bslbf
entry_point_flag	1	bslbf
if (entry_point_flag == 1) {		
auto_start_flag	1	bslbf
document_resolution	4	bslbf
use_xml	1	bslbf
default_version_flag	1	bslbf
independent_flag	1	bslbf
style_for_tv_flag	1	bslbf
reserved_future_use	4	bslbf
if (default_version_flag == 0) {		
bml_major_version	16	uimsbf
bml_minor_version	16	uimsbf
if (use_xml == 1) {		
bxml_major_version	16	uimsbf
bxml_minor_version	16	uimsbf
}		
}		
} else {		
reserved_future_use	5	bslbf
}		
if (transmission_format == '00') {		
additional_arib_carousel_info()		
ondemand_retrieval_flag	1	bslbf
file_storable_flag	1	bslbf
reserved_future_use	6	bslbf
}		
}		

Semantics of additional_arib_bxml_info():

transmission_format (Transmission Format):

This 2-bit field specifies the transmission method of BML/B-XML contents which is being transmitted within the corresponding contents.

Value	Semantics
00	Data carousel and event message transmission methods
01-11	Reserved for future use

entry_point_flag (Entry Point Flag):

This 1-bit flag indicates whether or not the component stream includes a BML/B-XML document that must be invoked first.

Value	Semantics
0	The component stream does not transmit a BML/B-XML document to be invoked.
1	The component stream transmits a BML/B-XML document to be invoked.

auto_start_flag (Auto Start Flag):

This flag indicates whether or not the BML/B-XML document should be invoked immediately.

Value	Semantics
0	The first BML/B-XML document is not invoked immediately.
1	The first BML/B-XML document is invoked immediately.

document_resolution (Document Resolution):

This value indicates resolution (corresponds to resolution property) and aspect ratio (corresponds to display-aspect-ratio property) of a BML/B-XML content. It has one of the following values.

Value	Semantics
0000	BML/B-XML documents with different resolutions and aspect ratios. This value also applicable to BML/B-XML documents with no resolutions nor aspect ratios specified.
0001	1920x1080 (16:9)
0010	1280x720 (16:9)
0011	960x540 (16:9)
0100	720x480 (16:9)
0101	720x480 (4:3)
0110	320x240 (4:3)
0111-1110	Reserved for future use
1111	This value is applicable to only BML/B-XML documents with no resolutions nor aspect ratios specified.

use_xml (XML Usage Flag):

This 1-bit flag indicates whether or not XML content that uses application specific tags is transmitted within the corresponding program.

Value	Semantics
0	XML content that uses application specific tags is not transmitted.
1	XML content that uses application specific tags is transmitted.

default_version_flag (Default Version Flag):

This 1-bit flag indicates whether or not to use default values of the major version and minor version numbers of BML and B-XML that is transmitted with the ES, which is defined in an operational standard regulation.

Value	Semantics
0	Does not use default values for version numbers.
1	Use default values for version numbers.

independent_flag (Independently Viewing Support Flag):

This flag indicates whether or not it is assumed that a data broadcasting program is deployed only in the corresponding program.

Value	Semantics
0	Independently Viewing is not supported.
1	Independently Viewing is supported.

style_for_tv_flag (Style Flag for TV display):

This flag indicates whether or not a content contained in the event has a style specification for "tv".

Value	Semantics
0	It contains only the contents which do not have tv for style and cannot be formatted on TV display.
1	It has tv for style and all the content can be formatted on TV display.

bml_major_version (BML Major Number):

This 16-bit field indicates the major number of the BML version that the BML browser must support to receive the contents.

bml_minor_version (BML Minor Number):

This 16-bit field indicates the minor number of the BML version that the BML browser must support to receive the contents.

bxml_major_version (B-XML Major Number):

This 16-bit field indicates the major number of the B-XML system version that the XML processor must support to receive the contents.

bxml_minor_version (B-XML Minor Number):

This 16-bit field indicates the minor number of the B-XML system version that the XML processor must support to receive the contents.

additional_arib_carousel_info():

Data structure defined in ARIB STD-B24 Volume 3, Annex C.1.

ondemand_retrieval_flag (On-demand Retrieval Support Flag):

This 1-bit area indicates whether or not the contents can be obtained from the carousel on-demand by user operation when receiving the contents transmitted with the ES. Detailed conditions are defined in an operational standard regulation for each transmission media.

Value	Semantics
0	On-demand retrieval is not supported.
1	On-demand retrieval is supported.

file_storable_flag (File Storage Support Flag): This flag indicates whether file storage of the data broadcasting program is allowed. For example, when the information is updated or the event start time is referred during the event, it is considered to be difficult to store the file. The detailed conditions whether it is supported or not are defined in an operational standard regulation for each transmission media.

9.3.3 Information encoded in selector area of data contents descriptor

An arib_bxml_info() structure shown in Table 9-9 must be described in the selector area of a Data Contents Descriptor.

Table 9-13 arib_bxml_info()

Syntax	Number of bits	Mnemonic
arib_bxml_info(){		
transmission_format	2	bslbf
reserved_future_use	1	bslbf
auto_start_flag	1	bslbf
document_resolution	4	bslbf
use_xml	1	bslbf
default_version_flag	1	bslbf
independent_flag	1	bslbf
content_id_flag	1	bslbf
associated_contents_flag	1	bslbf
reserved_future_use	1	bslbf
style_for_tv_flag	1	bslbf
update_flag	1	bslbf
ISO_639_language_code	24	bslbf
if (content_id_flag==1) {		
content_id	32	uimsbf
content_version	16	uimsbf
}		
if (default_version_flag==0) {		
bml_major_version	16	uimsbf
bml_minor_version	16	uimsbf

if(use_xml == 1) { bxml_major_version bxml_minor_version }	16 16	uimbsf uimbsf
if(transmission_format == '00') { arib_carousel_info() ondemand_retrieval_flag file_storable_flag reserved_future_use }	1 1 6	bslbf bslbf bslbf

Semantics of arib_bxml_info()

transmission_format (Transmission Format):

This 2-bit area specifies a transmission method of a BML/B-XML content that is transmitted with the corresponding contents.

Value	Semantics
00	Data carousel and event message transmission methods
01-11	Reserved for future use

auto_start_flag (Auto Start Flag):

This flag indicates whether or not to immediately invoke the BML/B-XML document.

Value	Semantics
0	Does not immediately invoke the first BML/B-XML document.
1	Immediately invokes the first BML/B-XML document.

document_resolution (Document Resolution):

This value indicates resolution and display-aspect-ratio of a BML/B-XML content. It has one of the following values.

Value	Semantics
0000	BML/B-XML documents with different resolutions and aspect ratios. This value also applicable to BML/B-XML documents with no resolutions nor aspect ratios specified.
0001	1920x1080 (16:9)
0010	1280x720 (16:9)
0011	960x540 (16:9)
0100	720x480 (16:9)
0101	720x480 (4:3)
0110	320x240 (4:3)
0111-1110	Reserved for future use
1111	This value is applicable to only BML/B-XML documents with no resolutions nor aspect ratios specified.

use_xml (XML Usage Flag):

This 1-bit flag indicates whether or not to transmit XML contents that use application specific tags with the corresponding program.

Value	Semantics
0	Does not transmit XML contents that use application specific tags.
1	Transmits XML contents that use application specific tags.

default_version_flag (Default Version Flag):

This 1-bit flag indicates whether or not to use default values for the major

version and minor version numbers of BML and B-XML that is transmitted within the ES, which is defined in an operational standard regulation.

Value	Semantics
0	Does not use default values for version numbers.
1	Use default values for version numbers.

independent_flag (Independently Viewing Support Flag):

This flag indicates whether or not it is assumed that a data broadcasting program is deployed only in the corresponding program.

Value	Semantics
0	Single reception is not supported.
1	Single reception is supported.

content_id_flag (Contents Identification Encoding Flag):

This flag indicates whether or not a data broadcasting program is assumed to be deployed using only the content in this event. (in other words, the content requires neither any data nor stream delivered in other events)

Value	Semantics
0	The content identification and content version are not included.
1	The content identification and content version are included.

associated_contents_flag (Associated Contents Flag):

If the content specified by this descriptor is an accompanied data service for a TV or radio program, this 1-bit flag indicates whether the content can work only with the associated TV or radio stream. It is always 0 for those contents which are not an accompanied data services.

Value	Semantics
0	The content is not an accompanied data service, or can work without TV/radio stream.
1	The content is an accompanied data service and can work only with TV/radio stream.

style_for_tv_flag (Style Flag for TV display):

This flag indicates whether or not a content contained in the event has a style specification for "tv".

Value	Semantics
0	It contains only the contents which do not have tv for style and cannot be formatted on TV display.
1	It has tv for style and all the content can be formatted on TV display.

update_flag (Update Flag):

This flag indicates whether or not there will be difference distribution for this content.

Value	Semantics
0	There will be no difference distribution for this B-XML/BML content.
1	There will be difference distribution for this B-XML/BML content.

ISO_639_language_code (Language Code):

This code indicates the language used in a BML contents.

content_id (Contents Identification):

This 32-bit field is a label that identifies a data broadcasting program. It is uniquely assigned within a scope of a broadcaster. When storing a data broadcasting event, if content_id has not been changed since the previous event, the new data can be overwritten.

content_version (Contents Version):

This 16-bit field indicates the version numbers of BML contents that have the same contents identification.

bml_major_version (BML Major Number):

This 16-bit field indicates the major number of the BML version that the BML browser must support to receive the contents.

bml_minor_version (BML Minor Number):

This 16-bit field indicates the minor number of the BML version that the BML browser must support to receive the contents.

bxml_major_version (B-XML Major Number):

This 16-bit field indicates the major number of the B-XML system version that the XML processor must support to receive the contents.

bxml_minor_version (B-XML Minor Number):

This 16-bit field indicates the minor number of the B-XML system version that the XML processor must support to receive the contents.

arib_carousel_info(): Data structure defined in ARIB STD-B24 Volume 3, Annex C.

ondemand_retrieval_flag (On-demand Retrieval Support Flag):

This 1-bit area indicates whether or not the contents can be obtained from the carousel on-demand by user operation when receiving the contents transmitted with the ES. Detailed conditions are defined in an operational standard regulation for each transmission media.

Value	Semantics
0	On-demand retrieval is not supported.
1	On-demand retrieval is supported.

file_storable_flag (File Storage Support Flag):

This flag indicates whether file storage of the data broadcasting program is supported or not. File storage is difficult in the specific cases, for example, where the information is updated during program events and where the time related to the actual broadcasting schedule is referenced. The detailed conditions for support are defined for each media.

9.3.4 Information to be written in private area of DII

When the data coding scheme identification is B-XML/BML, an arib_bxml_privatedata_descriptor() structure shown in 9-10 is written as a descriptor in the private area of DII.

Table 9-14 arib_bxml_privatedata_descriptor()

Syntax	Number of bits	Mnemonic
arib_bxml_privatedata_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
return_to_entry_flag	1	bslbf
Reserved	7	bslbf
}		

Semantics of arib_bxml_privatedata_descriptor()

descriptor_tag: Tag value of `arib_bxml_privatedata_descriptor()`. It is set to 0xF0.

return_to_entry_flag (Forced Return Flag):
If this flag is 1, a BML document which is destined to be invoked before any other document in this component is forced to be invoked first on an update of data event, no matter whether a BML/B-XML document of other components is being watched. Any destined, that is, start-up BML/B-XML document complies with the convention defined in Section 9.2.2.

9.3.5 Descriptor in the module information area of DII

The descriptor in the module information area of DII conforms to the definition in ARIB STD-B24 Volume 3. Table 9-11 shows data structure of `control_data_byte()` which is included in the Control descriptor. If the BML version number and B-XML version number of a content included in the module are the same as those in the additional information area of the Data Component Descriptor of the component that carries the module specified by PMT, the Control descriptor may not be encoded.

Table 9-15 Data structure of `control_data_byte()` of Control Descriptor

Syntax	Number of bits	Mnemonic
<code>control_data_byte(){</code>		
<code>use_xml</code>	1	bslbf
<code>reserved</code>	7	
<code>bml_major_version</code>	16	uimsbf
<code>bml_minor_version</code>	16	uimsbf
<code>if (use_xml == 1) {</code>		
<code>bxml_major_version</code>	16	uimsbf
<code>bxml_minor_version</code>	16	uimsbf
<code>}</code>		
<code>}</code>		

Semantics of `control_data_byte()`

`use_xml`

(XML Usage Flag):

This 1-bit flag indicates whether the module includes XML contents that use application specific tags.

Value	Semantics
0	The module includes XML contents that use application specific tags.
1	The module does not include XML contents that use application specific tags.

`bml_major_version`(BML Major Number):

A 16-bit major number of the BML format on that the BML contents included in the module are based.

`bml_minor_version`(BML Minor Number):

A 16-bit minor number of the BML format on that the BML contents included in the module are based.

`bxml_major_version`(B-XML Major Number):

A 16-bit major number of the BML system on that the B-XML contents included in the module are based.

`bxml_minor_version`(B-XML Minor Number):

A 16-bit minor number of the BML system on that the B-XML contents included in the module are based.

Chapter 10 XHTML-based BML Encoding using XML Namespace

10.1 XML Namespace

The conventions on the BML namespace complies with the W3C Recommendation, “Namespaces in XML 14-January-1999(REC-xml-names-1999014), (JIS-TR X 0023:1999)”.

This section defines the additional conventions on the BML namespace.

- An XML namespace prefix must begin with the string “bml”, which is case-insensitive.
- The scope of the id element is an entire BML document. The id element must be unique in an entire BML document and must not be localized by the namespace. That is, the id element of an element defined with xhtml and the id element of an element extended with bml are unique in the BML document.
- A tag name in a CSS selector must not contain an XML namespace prefix.

10.2 BML Encoding and XML Namespace

To describe a BML document without using the XML namespace, the following rules are used.

- The document declaration must be “bml”.
- The root element must be “bml”.
- The DTD must be a formal public identifier (FPI) whose value is “-//ARIB//DTD BML x.y//JA”, where x and y constitute the version number of DTD.

For example:

```
<?xml version="1.0" encoding="EUC-JP" ?>
<!DOCTYPE bml PUBLIC
    "-//ARIB//DTD BML x.y//JA" http://www.arib.or.jp/B24/DTD/bml_x_y.dtd">
<?bml bml-version="a.b" ?>
<bml>
<head>
<title>This is a sample document.</title>
<script>...omitted...</script>
</head>
<body style="resolution:960x540; used-key-list:data-button;" onload="l();">
...omitted...
</body>
</bml>
```

To describe a BML document by using the XML namespace into a XHTML-based format, the following rules are used:

- The document declaration must be “html”.
- The root element must be “html”.
- The XML namespace prefix must be “bml”.

- The DTD must be a formal public identifier (FPI) whose value is "-//ARIB//DTD XHTML BML Profile-name x.y//JA ", where Profile-name must be a unique name of a profile, and x and y constitute the version number of DTD.
- The value of the xmlns attribute of the html root element must be "http://www.w3.org/1999/xhtml1".

For example:

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<!DOCTYPE html PUBLIC "-//ARIB//DTD XHTML BML x.y//JA" "bml_x_yb.dtd">
<?bml bml-version="a.b" ?>
<html xmlns="http://www.w3.org/1999/xhtml1">
<head>
<title>This is a sample document.</title>
<script>...omitted...</script>
<link rel="stylesheet" type="text/css" href="default.css" />
</head>
<body style="resolution:960x540; used-key-list:data-button;" onload="l();">
...omitted...
</body>
</html>
```

It is recommended that the available media type to a BML document are defined as the following:

- To specify a BML document that is based on XHTML, and have no extension module defined in Section 5.3.20, no extended object for broadcasting, nor no extended function for broadcasting, "text/html" must be used.
- Otherwise, "text/X-arib-bml" must be used.

Annex A Coding Schemes of Color Map Data

Color map data applicable to the clut property is encoded using the data structure defined in Table A-1.

Table A-1 Encoding of Color Map Data

Syntax	Bits	Mnemonic
color_map_data {		
clut_type	1	bslbf
Depth	2	bslbf
region_flag	1	bslbf
start_end_flag	1	bslbf
reserved_future_use	3	bslbf
if (region_flag == 1) {		
top_left_x	16	uimsbf
top_left_y	16	uimsbf
bottom_right_x	16	uimsbf
bottom_right_y	16	uimsbf
}		
if (start_end_flag == 1) {		
if (depth == '00') {		
start_index	4	uimsbf
end_index	4	uimsbf
} else if (depth == '01') {		
start_index	8	uimsbf
end_index	8	uimsbf
} else if (depth == '10') {		
start_index	16	uimsbf
end_index	16	uimsbf
}		
for (i=start_index; i<=end_index; i++) {		
if (type == 0) {		
Y	8	uimsbf
CB	8	uimsbf
CR	8	uimsbf
} else {		
R	8	uimsbf
G	8	uimsbf
B	8	uimsbf
}		
Alpha	8	uimsbf
}		
} else {		
for (i=0; i<N(Note); i++) {		
if (type == 0) {		
Y	8	uimsbf
CB	8	uimsbf
CR	8	uimsbf
} else {		
R	8	uimsbf
G	8	uimsbf
B	8	uimsbf
}		
Alpha	8	uimsbf

<pre> } } } </pre>		
--------------------	--	--

Note: N indicates the maximum number of CLUTs specified with depth.

Semantics of color_map_data()

type (Color Map Type):

This 1-bit field indicates a color space used for specifying color map data in the data structure.

Value of type	Semantics
0	Specifies with YCBCR color space.
1	Specifies with RGB color space.

The relationship between YCBCR and RGB conforms to ARIB STD-B24 Volume 1, Section 1.

depth (Depth Specification):

This 2-bit field specifies the depth of CLUT where color map data is set.

Value of depth	Semantics
00	4-bit CLUT (Maximum 16 colors)
01	8-bit CLUT (Maximum 256 colors)
10	16-bit CLUT (Maximum 65536 colors)
11	Reserved

region_flag (Region Specification Flag):

This 1-bit field indicates, given that CLUT contains the Color Map data has the capability of being applied to each specified sub region in a screen, whether the region specified is encoded in the data structure or not.

Value of region_flag	Semantics
0	Does not set region CLUT and not encode region specification.
1	Sets region CLUT and encodes region specification.

start_end_flag (Start/End Index Flag):

This 1-bit field indicates whether or not the following color map specification is for all CLUT elements. If it is not for all CLUT elements, the start index and stop index of CLUT to be set are encoded.

Value of start_end_flag	Semantics
0	Does not specify start/end indexes. The data structure has a color map data for all CLUT elements.
1	Specifies star/end indexes.

start_index (Start Index):

The length of this field depends on the specification of depth. This field indicates an index number of the first element in CLUT to that the color map is specified.

end_index (End Index):

The length of this field depends on the specification of depth. This field indicates an index number of the last element in CLUT to that the color map is specified.

top_left_x, top_left_y, bottom_right_x, and bottom_right_y (Region Specification):

These 16-bit fields indicate the top left x coordinate, top left y coordinate,

bottom right x coordinate, and bottom right y coordinate of a sub region on the screen where the color map data is set.

Y, CB, CR, R, G, and B (Color Specification):

These are 8-bit fields. A series of three fields indicate an element of color map data used for specifying the color space for each type specification.

alpha (Translucency):

This 8-bit field indicates the translucency to be set with either YCBCR or RGB data to an element of CLUT.

Annex B Coding Schemes for Designation of Regions Using Zip Code

B.1 Overall Structure

An encoded region designation has one or more sequences of the following data structure(s).

length 8-bit: Size of list in bytes
exclude_list_length 8-bit: Size of exclude list in bytes
exclude_list
include_list

If exclude_list_length is 0, there is no exclude_list. If a zip code to be retrieved is in the exlude_list, it is excluded. For a zip code to be retrieved, it must not be in exclude_list and it must be in include_list.

B.2 Base Format

The base format is a repetition of a 32-bit fixed format.

If the MSB is 0, the range is specified in 2 digits with start and end. If the MSB of the following byte is 1, another 2-bit field follows.

If the MSB is 1, the first 7 bits represents a 2-digit number. The following 4 bits divide the remaining 3 bytes and the first 1 nibble is used to switch the format of the field that follows.

“5 digit range From” and “5 digit range To,” and “7 digit range From” and “7 digit range To” must be placed side by side. Specification with less than 7 digits is interpreted as any zip code whose upper digits are equivalent to the specified digits. For example, specifying “30 to 35” matches all zip codes from 300-0000 to 359-9999.

Table B-1 Base Format

0	From	0	To	*	Void	*	Void
0	From	1	To	*	From	*	To
1	2digits	Flag	a	b	c	d	e

Table B-2 Flag and Format of Following Field

Flag		a-e
3digit list	0x8	0,1,2,3,4,5,6,7,8,9,F(void)
3digit range	0x9	a:F(void) b:From,c:To d:From e:To
5digit list	0xA	a:3rd digit b:4th digit c:5th digit d:4th digit e:5th digit
5digit range From	0xB	a:3rd digit b:4th digit c:5th digit d,e:void
5digit range To	0xC	a:3rd digit b:4th digit c:5th digit d,e:void
7digit range From	0xD	a:3rd digit b:4th digit c:5th digit d:6th digit e:7th digit
7digit range To	0xE	a:3rd digit b:4th digit c:5th digit d:6th digit e:7th digit
7digit list	0xF	a:3rd digit b:4th digit c:5th digit d:6th digit e:7th digit

- (1) Using the 3digit list, maximum of 5 sets per 4 bytes can be designated. When less than 5 sets are designated, the corresponding field(s) should be set to F(void).
- (2) Using the 5digit list, maximum of 2 sets per 4 bytes can be designated. When only one set is designated, the ‘d’ and ‘e’ fields should be set to F(void).
- (3) Using the 3digit range, maximum of 2 sets per 4 bytes can be designated.

(4) In the case of 3digit list, the fields from ‘a’ to ‘e’ should have the last digit of the 3digit.

B.3 Examples

Kanto Koshin’etsu: 10 to 40, 94 to 95 (Zip codes 100-0000 to 409-9999 and 940-0000 to 959-9999)

5		0					
0	10	1	40	0	94	0	95

Tohoku: 01 to 03, 96 to 99

5		0					
0	01	1	03	0	96	0	99

Hokkaido: 00, 04 to 09

5		0					
0	00	1	00	0	04	0	09

Kanto: 10 to 37, 384-0097, 389-0121

13		0					
0	10	0	37	0	0x7F	0	0x7F
1	38	0xF	4	0	0	9	7
1	38	0xF	9	0	1	2	1

Tokyo: 10 to 20, not 199

9		4					
1	19	0x8	9	0xF	0xF	0xF	0xF
0	10	0	20	0	0x7F	0	0x7F

Osaka: 53 to 59, 618-0000 to 618-5000, 630-0271, not 563-0801 (618-0000 includes Kyoto-fu.)

21		4					
1	56	0xF	3	0	8	0	1
0	53	1	59	0	0x7F	0	0x7F
1	61	0xD	8	0	0	0	0
1	61	0xE	8	5	0	0	0
1	63	0xF	0	0	2	7	1

- The following data are not allowed.

- * 3digit list
a:1 b:(void): c:3 d:4 e:5
- * 3digit range
a:(void) b:(void): c:(void) d:From e:To
a:(void) b:(void): c:To d:From e:(void)
- * 5digit list
a:3rd b:(void): c:(void) d:4th e:5th

Annex C Media Type of B-XML/BML Documents and Monomedia Data

Table C-1 defines the available media types to type attribute of object element in a BML document, and type descriptor of DII and their semantics. These media types conform to RFC2046. However, “Schema” column in the table C-1 indicates the schema used to transmit data carousels in broadcast channel.

Among the extended media types that start with ‘X’, those that does not start with ”X-arib” are to be proprietarily defined for an individual media, or by a broadcaster, terminal manufacturer and other consortiums. To prevent from media types described above conflicting with each other, additional conventions on the media types are defined separately from this standard in an operational rule.

Table C-1 Media Types and Semantics

Schema	Media Type	Semantics
arib-dc: (Data carousel)	multipart/mixed	Multipart type
	text/xml; charset="euc-jp"	XML document (EUC)
	text/xml; charset="UTF-16"	XML document (UTF-16)
arib-file: (Strage media)	text/xml; charset="Shift JIS"	XML document (Shift JIS)
	text/css	CSS File
http:,https:, ftp:	text/xsl; charset="euc-jp"	XSL document (EUC)
	text/xsl; charset="UTF-16"	XSL document (UTF-16)
	text/xsl; charset="Shift JIS"	XSL document (Shift JIS)
	text/html; charset="euc-jp"	XHTML 1.0 compliant BML document (EUC) without extended functions
	text/html; charset="UTF-16"	XHTML 1.0 compliant BML document (UTF-16) without extended functions
	text/html; charset="Shift JIS"	XHTML 1.0 compliant BML document (Shift JIS) without extended functions
	text/X-arib-bml; charset="euc-jp"	BML document (EUC)
	text/X-arib-bml; charset="UTF-16"	BML document (UTF-16)
	text/X-arib-bml; charset="Shift JIS"	BML document (Shift JIS)
	text/plain ; charset="euc-jp"	Plane text (EUC)
	text/plain ; charset="UTF-16"	Plane text (UTF-16)
	text/plain; charset="Shift JIS"	Plane text (Shift JIS)
	text/X-arib-jis8text	Text with control codes (8-bit code)
	text/X-arib-ecmascript; charset="euc-jp"	ECMAScript (EUC)
	text/X-arib-ecmascript; charset="UTF-16"	ECMAScript (UTF-16)
	text/X-arib-ecmascript; charset="Shift JIS"	ECMAScript (Shift JIS)
	application/X-arib-meta+xml; charset="UTF-8"	Metadata represented in XML(UTF-8)
	application/X-arib-meta+xml; charset="UTF-16"	Metadata represented in XML (UTF-16)
	image/jpeg	JPEG
	image/png	PNG
	image/gif	GIF

Schema	Media Type	Semantics
	image/X-arib-png	PNG subset (without PLTE chunk)
	image/X-arib-mng	MNG subset
	image/X-arib-mpeg2-I	MPEG-2-I frame
	image/X-arib-mpeg4-I-simple	MPEG-4-I-VOP (simple profile)
	image/X-arib-mpeg4-I-core	MPEG-4-I-VOP (core profile)
	image/X-arib-H264-I-baseline	H.264 MPEG-4 AVC I-picture (baseline profile)
	image/X-arib-H264-I-main	H.264 MPEG-4 AVC I-picture (main profile)
	audio/X-arib-mpeg2-aac	MPEG-2 AAC
	audio/X-arib-mpeg2-bc	MPEG-2 BC
	audio/X-arib-aiff	PCM(AIFF-C)
	audio/X-arib-additional	Additional sound
	video/X-arib-mpeg1	MPEG-1 system stream
	application/xhtml+xml; charset="euc-jp"	XHTML-Print document (EUC)
	application/xhtml+xml; charset="Shift JIS"	XHTML-Print document (Shift JIS)
	application/X-arib-stream-text; charset="euc-jp"	Plane text (EUC) updated by version up of data carousel
	application/X-arib-stream-text; charset="UTF-16"	Plane text (UTF-16) updated by version up of data carousel
	application/X-arib-stream-text; charset="Shift JIS"	Plane text (Shift JIS) updated by version up of data carousel
	application/X-arib-stream-jis8text	Text with control codes (8-bit code) updated by version up of data carousel
	application/X-arib-stream-png	PNG subset updated by version up of data carousel
	application/X-arib-stream-jpeg	JPEG subset updated by version up of data carousel
	application/X-arib-stream-mpeg2-I	MPEG-2-I frame updated by version up of data carousel
	application/X-arib-stream-mpeg4-I-simple	MPEG-4-I-VOP (simple profile) updated by version up of data carousel
	application/X-arib-stream-mpeg4-I-core	MPEG-4-I-VOP (core profile) updated by version up of data carousel
	application/X-arib-stream-H264-I-baseline	H.264 MPEG-4 AVC I-picture (baseline profile) updated by version up of data carousel
	application/X-arib-stream-H264-I-main	H.264 MPEG-4 AVC I-picture (main profile) updated by version up of data carousel
	application/X-arib-mpeg2-ts	MPEG-2 transport stream
	application/X-arib-mpeg2-tts	Time-stamped TS format file
	application/X-arib-bmlclut	CLUT file used to display a BML document
	application/X-arib-btable	Binary Table
	application/X-arib-drcs	DRCS
	application/X-arib-PDI	PDI (geometric)-revised
	application/X-arib-resourceList	Resource information included in the module

Schema	Media Type	Semantics
	application/X-arib-storedResourceList	Stored resource list
	application/X-arib-rootcertificate	Root certificate
arib: (PES)	audio/X-arib-mpeg2-aac	MPEG2-AAC
	audio/X-arib-mpeg2-bc	MPEG2-BC
	audio/X-arib-mpeg4	MPEG-4 audio stream (audio PES)
	video/X-arib-mpeg1	MPEG-1 video stream
	video/X-arib-mpeg2	MPEG-2 video stream
	video/X-arib-mpeg4-simple	MPEG-4 video stream (simple profile)
	video/X-arib-mpeg4-core	MPEG-4 video stream (core profile)
	video/X-arib-H264-baseline	H.264 MPEG-4 AVC video stream (baseline profile)
	video/X-arib-H264-main	H.264 MPEG-4 AVC video stream (main profile)
arib-ic: (Still image carousel)	image/X-arib-mpeg2-I	MPEG-2-I frame
	image/X-arib-mpeg4-I-simple	MPEG-4-I-VOP (simple profile)
	image/X-arib-mpeg4-I-core	MPEG-4-I-VOP (core profile)
	application/X-arib-stream-mpeg2-I	MPEG-2-I frame updated by version up of still image carousels
	application/X-arib-stream-mpeg4-I-simple	MPEG-4-I-VOP (simple profile) updated by version up of still image carousels
	application/X-arib-stream-mpeg4-I-core	MPEG-4-I-VOP (core profile) updated by version up of still image carousels
	application/X-arib-stream-H264-I-baseline	H.264 MPEG-4 AVC I-picture (baseline profile) updated by version up of still image carousels
	application/X-arib-stream-H264-I-main	H.264 MPEG-4 AVC I-picture (main profile) updated by version up of still image carousels
romsound:	audio/X-arib-romsound	Rom sound built in the receiver

Annex D Document Type Definition of BML

This section defines the DTDs (Document Type Definitions) used in this specification. This specification uses the DTDs extended from XHTML 1.0 Strict DTD.

D.1 BML Driver DTD

```
<!-- ..... -->
<!-- BML 2.0 DTD ..... -->
<!-- file: bml_2_0.dtd
-->

<!-- BML 2.0 DTD

    This is BML, a broadcast markup language as a modular XML application.
    Copyright 2006 ARIB, All Rights Reserved.
    Revision: $Id: bml_2_0.mod,v 1.1 2006/03/01 00:00 ARIB Data Broadcasting
    WG PE-TG

-->

<!-- This is the driver file for version 2.0 of the BML DTD.

    Please use this formal public identifier to identify it:

        "-//ARIB//DTD XHTML BML 2.0//JA"
-->
<!ENTITY % XHTML.version "-//ARIB//DTD XHTML BML 2.0//JA" >

<!-- Use this URI to identify the default namespace:

    "http://www.w3.org/1999/xhtml"

    See the Qualified Names module for information
    on the use of namespace prefixes in the DTD.
-->
<!ENTITY % NS.prefix "INCLUDE" >
<!ENTITY % XHTML.prefix "" >

<!-- Reserved for use with the XLink namespace:
-->
<!ENTITY % XLINK.xmlns "" >
<!ENTITY % XLINK.xmlns.attrib "" >

<!-- reserved for future use with document profiles -->
<!ENTITY % XHTML.profile "" >

<!-- Bring in any qualified name modules outside of XHTML -->
<!ENTITY % bml-qname.mod
    PUBLIC "-//W3C//ENTITIES BML Qualified Names 2.0//JA"
        "http://www.arib.or.jp/B24/DTD/bml-qname_2_0.mod" >
%bml-qname.mod;
```

```
<!-- Bidirectional Text features
      This feature-test entity is used to declare elements
      and attributes used for bidirectional text support.
-->
<!ENTITY % XHTML.bidi  "INCLUDE" >

<!-- Pre-Framework Redeclaration placeholder ..... -->
<!-- this serves as a location to insert markup declarations
      into the DTD prior to the framework declarations.
-->
<!ENTITY % LanguageCode.datatype "NMTOKEN" >
<!ENTITY % lang.attrib
      "xml:lang      %LanguageCode.datatype;  #IMPLIED"
>

<![%XHTML.bidi;[
<!ENTITY % dir.attrib
      "dir          ( ltr | rtl )          #IMPLIED"
>

<!ENTITY % orientation.attrib
      "%BML.pfx;orientation      ( vert | horiz )      #IMPLIED"
>

<!ENTITY % I18n.attrib
      "%dir.attrib;
      %orientation.attrib;
      %lang.attrib;"
>
<!ENTITY % xhtml-prefw-redecl.module "IGNORE" >
<![%xhtml-prefw-redecl.module;[
%xhtml-prefw-redecl.mod;
<!-- end of xhtml-prefw-redecl.module -->]]>

<!ENTITY % xhtml-events.module "INCLUDE" >

<!-- Inline Style Module ..... -->
<!ENTITY % xhtml-inlstyle.module "INCLUDE" >
<![%xhtml-inlstyle.module;[
<!ENTITY % xhtml-inlstyle.mod
      PUBLIC "-//W3C//ELEMENTS XHTML Inline Style 1.0//EN"
      "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-inlstyle-1.mod" >
%xhtml-inlstyle.mod;]]>

<!-- declare Document Model module instantiated in framework
-->
<!ENTITY % xhtml-model.mod
      PUBLIC "-//W3C//ENTITIES BML DOCUMENT MODEL 2.0//JA"
      "http://www.arib.or.jp/B24/DTD/bml-model_2_0.mod" >

<!-- Modular Framework Module (required) ..... -->
<!ENTITY % xhtml-framework.module "INCLUDE" >
<![%xhtml-framework.module;[
```

```
<!ENTITY % xhtml-framework.mod
    PUBLIC "-//W3C//ENTITIES XHTML Modular Framework 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-framework-1.mod" >
%xhtml-framework.mod;]]>

<!-- Post-Framework Redeclaration placeholder ..... -->
<!-- this serves as a location to insert markup declarations
    into the DTD following the framework declarations.
-->
<!ENTITY % xhtml-postfw-redecl.module "IGNORE" >
<![%xhtml-postfw-redecl.module;[
%xhtml-postfw-redecl.mod;
<!-- end of xhtml-postfw-redecl.module -->]]>

<!-- Text Module (Required) ..... -->
<!ENTITY % xhtml-text.module "INCLUDE" >
<![%xhtml-text.module;[
<!ENTITY % xhtml-text.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Text 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-text-1.mod" >
%xhtml-text.mod;]]>

<!-- Hypertext Module (required) ..... -->
<!ENTITY % xhtml-hypertext.module "INCLUDE" >
<![%xhtml-hypertext.module;[
<!ENTITY % xhtml-hypertext.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Hypertext 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-hypertext-1.mod" >
%xhtml-hypertext.mod;]]>

<!-- Lists Module (required) ..... -->
<!ENTITY % xhtml-list.module "INCLUDE" >
<![%xhtml-list.module;[
<!ENTITY % xhtml-list.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Lists 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-list-1.mod" >
%xhtml-list.mod;]]>

<!-- ..... -->

<!-- Edit Module ..... -->
<!ENTITY % xhtml-edit.module "INCLUDE" >
<![%xhtml-edit.module;[
<!ENTITY % xhtml-edit.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Editing Elements 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-edit-1.mod" >
%xhtml-edit.mod;]]>

<!-- BIDI Override Module ..... -->
<!ENTITY % xhtml-bdo.module "%XHTML.bidi;" >
<![%xhtml-bdo.module;[
<!ENTITY % xhtml-bdo.mod
    PUBLIC "-//W3C//ELEMENTS XHTML BIDI Override Element 1.0//EN"
```

```
"http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-bdo-1.mod" >
%xhtml-bdo.mod;]]>

<!-- Ruby Module ..... -->
<!ENTITY % Ruby.common.attlists "IGNORE" >
<!ENTITY % Ruby.common.attrib "%Common.attrib;" >
<!ENTITY % xhtml-ruby.module "IGNORE" >
<![%xhtml-ruby.module;[
<!ENTITY % xhtml-ruby.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Ruby 1.0//EN"
        "http://www.w3.org/TR/ruby/xhtml-ruby-1.mod" >
%xhtml-ruby.mod;]]>

<!-- Presentation Module ..... -->
<!ENTITY % xhtml-pres.module "INCLUDE" >
<![%xhtml-pres.module;[
<!ENTITY % xhtml-pres.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Presentation 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-pres-1.mod" >
%xhtml-pres.mod;]]>

<!-- Link Element Module ..... -->
<!ENTITY % xhtml-link.module "INCLUDE" >
<![%xhtml-link.module;[
<!ENTITY % xhtml-link.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Link Element 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-link-1.mod" >
%xhtml-link.mod;]]>

<!-- Document Metainformation Module ..... -->
<!ENTITY % xhtml-meta.module "INCLUDE" >
<![%xhtml-meta.module;[
<!ENTITY % xhtml-meta.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Metainformation 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-meta-1.mod" >
%xhtml-meta.mod;]]>

<!-- Base Element Module ..... -->
<!ENTITY % xhtml-base.module "INCLUDE" >
<![%xhtml-base.module;[
<!ENTITY % xhtml-base.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Base Element 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-base-1.mod" >
%xhtml-base.mod;]]>

<!-- Scripting Module ..... -->
<!ENTITY % xhtml-script.module "INCLUDE" >
<![%xhtml-script.module;[
<!ENTITY % xhtml-script.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Scripting 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-script-1.mod" >
%xhtml-script.mod;]]>
```

```
<!-- Style Sheets Module ..... -->
<!ENTITY % xhtml-style.module "INCLUDE" >
<![%xhtml-style.module;[
<!ENTITY % xhtml-style.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Style Sheets 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-style-1.mod" >
%xhtml-style.mod;]]>

<!-- Image Module ..... -->
<!ENTITY % xhtml-image.module "INCLUDE" >
<![%xhtml-image.module;[
<!ENTITY % xhtml-image.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Images 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-image-1.mod" >
%xhtml-image.mod;]]>

<!-- Client-side Image Map Module ..... -->
<!ENTITY % xhtml-csismap.module "INCLUDE" >
<![%xhtml-csismap.module;[
<!ENTITY % xhtml-csismap.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Client-side Image Maps 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-csismap-1.mod" >
%xhtml-csismap.mod;]]>

<!-- Server-side Image Map Module ..... -->
<!ENTITY % xhtml-ssismap.module "INCLUDE" >
<![%xhtml-ssismap.module;[
<!ENTITY % xhtml-ssismap.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Server-side Image Maps 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-ssismap-1.mod" >
%xhtml-ssismap.mod;]]>

<!-- Param Element Module ..... -->
<!ENTITY % xhtml-param.module "INCLUDE" >
<![%xhtml-param.module;[
<!ENTITY % xhtml-param.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Param Element 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-param-1.mod" >
%xhtml-param.mod;]]>

<!-- Embedded Object Module ..... -->
<!ENTITY % xhtml-object.module "INCLUDE" >
<![%xhtml-object.module;[
<!ENTITY % xhtml-object.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Embedded Object 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-object-1.mod" >
%xhtml-object.mod;]]>

<!-- Tables Module ..... -->
<!ENTITY % xhtml-table.module "INCLUDE" >
<![%xhtml-table.module;[
<!ENTITY % xhtml-table.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Tables 1.0//EN"
```

```
"http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-table-1.mod" >
%xhtml-table.mod;]]>

<!-- Basic Tables Module ..... -->
<!ENTITY % xhtml-basic-table.module "IGNORE" >
<![%xhtml-basic-table.module;[
<!ENTITY % xhtml-basic-table.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Basic Tables 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-basic-table-1.mod"
>
%xhtml-basic-table.mod;]]>

<!-- Forms Module ..... -->
<!ENTITY % xhtml-form.module "INCLUDE" >
<![%xhtml-form.module;[
<!ENTITY % xhtml-form.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Forms 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-form-1.mod" >
%xhtml-form.mod;]]>

<!-- Basic Forms Module ..... -->
<!ENTITY % xhtml-basic-form.module "IGNORE" >
<![%xhtml-basic-form.module;[
<!ENTITY % xhtml-basic-form.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Basic Forms 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-basic-form-1.mod"
>
%xhtml-basic-form.mod;]]>

<!-- Legacy Markup ..... -->
<!ENTITY % xhtml-legacy.module "IGNORE" >
<![%xhtml-legacy.module;[
<!ENTITY % xhtml-legacy.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Legacy Markup 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-legacy-1.mod" >
%xhtml-legacy.mod;]]>

<!-- Frames Module ..... -->
<!ENTITY % xhtml-frames.module "INCLUDE" >
<![%xhtml-frames.module;[
<!ENTITY % xhtml-frames.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Frames 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-frames-1.mod" >
%xhtml-frames.mod;]]>

<!-- Target Markup ..... -->
<!ENTITY % xhtml-target.module "INCLUDE" >
<![%xhtml-target.module;[
<!ENTITY % target-iframe.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Target 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-target-1.mod" >
%xhtml-target.mod;]]>
```

```
<!-- IFrame Module ..... -->
<!ENTITY % xhtml-iframe.module "INCLUDE" >
<![%xhtml-iframe.module;[
<!ENTITY % xhtml-iframe.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Inline Frame Element 1.0//EN"
        "http://www.w3.org/TR/xhtml1-modularization/DTD/xhtml-iframe-1.mod" >
%xhtml-iframe.mod;]]>

<!-- Name Identification Markup ..... -->
<!ENTITY % xhtml-nameident.module "IGNORE" >
<![%xhtml-nameident.module;[
<!ENTITY % xhtml-nameident.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Name Identifier 1.0//EN"
        "http://www.w3.org/TR/xhtml1-modularization/DTD/xhtml-nameident-1.mod"
%xhtml-nameident.mod;]]>

<!-- Document Structure Module (required) ..... -->
<!ENTITY % xhtml-struct.module "INCLUDE" >
<![%xhtml-struct.module;[
<!ENTITY % xhtml-struct.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Document Structure 1.0//EN"
        "http://www.w3.org/TR/xhtml1-modularization/DTD/xhtml-struct-1.mod" >
%xhtml-struct.mod;]]>

<!-- ..... -->

<!-- BML Element Module ..... -->
<!ENTITY % bml-elements.module "INCLUDE" >
<![%bml-elements.module;[
<!ENTITY % bml-elements.mod
    PUBLIC "-//ARIB//ELEMENTS XHTML BML Elements 2.0//JA"
        "http://www.arib.or.jp/B24/DTD/bml-elements_2_0.mod" >
%bml-elements.mod;]]>

<!-- Basic BML Element Module ..... -->
<!ENTITY % bml-basic-elements.module "IGNORE" >
<![%bml-basic-elements.module;[
<!ENTITY % bml-basic-elements.mod
    PUBLIC "-//ARIB//ELEMENTS XHTML Basic BML Elements 2.0//JA"
        "http://www.arib.or.jp/B24/DTD/bml-basic-elements_2_0.mod" >
%bml-basic-elements.mod;]]>

<!-- Basic Mobile BML Element Module ..... -->
<!ENTITY % bml-basic-mobile-elements.module "IGNORE" >
<![%bml-basic-mobile-elements.module;[
<!ENTITY % bml-basic-mobile-elements.mod
    PUBLIC "-//ARIB//ELEMENTS XHTML Basic Mobile BML Elements 2.0//JA"
        "http://www.arib.or.jp/B24/DTD/bml-basic-mobile-elements_2_0.mod" >
%bml-basic-mobile-elements.mod;]]>

<!-- Server BML Element Module ..... -->
<!ENTITY % bml-server-elements.module "IGNORE" >
<![%bml-server-elements.module;[
<!ENTITY % bml-server-elements.mod
    PUBLIC "-//ARIB//ELEMENTS XHTML Server BML Elements 1.0//JA"
```

D.2 BML Extension Elements DTD

 $\langle | \dots \rangle$


```
<!-- bevent: BML Interruption Event Description Element -->

<!ENTITY % BML.bevent.content "(%BML.beitem.qname;)+ " >
<!ELEMENT %BML.bevent.qname; %BML.bevent.content; >
<!ATTLIST %BML.bevent.qname;
    %id.attrib;
>

<!-- beitem: BML Interruption Event Item Element -->

<!ENTITY % BML.beitemType.class
    "( EventMessageFired | EventFinished | EventEndNotice | Abort
      | ModuleUpdated | ModuleLocked | TransmissionFinished
      | TimerFired | DataEventChanged | CCStatusChanged
      | MainAudioStreamChanged | NPTReferred | MediaStarted
      | MediaStopped | MediaRepeated | DataButtonPressed
      | IPConnectionTerminated | PeripheralEventOccured | StoreFinished
      | DataEventChangedEx | SegmentPlayEnded | MetadataUpdated )"
>
<!ENTITY % BML.beitemTimeMode.class
    "( absolute | origAbsolute | relativeToEvent
      | relativeToLoad | NPT )"
>
<!ENTITY % BML.beitem.content "EMPTY" >
<!ELEMENT %BML.beitem.qname; %BML.beitem.content; >
<!ATTLIST %BML.beitem.qname;
    id ID #REQUIRED
    type %BML.beitemType.class; #REQUIRED
    onoccur %Script.datatype; #REQUIRED
    es_ref %URI.datatype; #IMPLIED
    message_id %Number.datatype; #IMPLIED
    message_version %Number.datatype; #IMPLIED
    message_group_id %Number.datatype; #IMPLIED
    module_ref %URI.datatype; #IMPLIED
    language_tag %Number.datatype; #IMPLIED
    register_id %Number.datatype; #IMPLIED
    service_id %Number.datatype; #IMPLIED
    event_id %Number.datatype; #IMPLIED
    peripheral_ref %URI.datatype; #IMPLIED
    time_mode %BML.beitemTimeMode.class; #IMPLIED
    time_value CDATA #IMPLIED
    object_id ID #IMPLIED
    segment_id ID #IMPLIED
    subscribe (subscribe) #IMPLIED
>

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!-- Defining additional attributes to body element. -->
<!ENTITY % BML.body.invisible.qname "%BML.pfx;invisible" >
<!ATTLIST %body.qname;
```

```

    %BML.body.invisible.qname;      (invisible)      #IMPLIED
>

<!-- Declaring additional attribute for div, p, span, object element -->
<!ENTITY % BML.attr.accesskey.qname "%BML.pfx;accesskey" >
<!ENTITY % BML.accesskey.attrib
    "%BML.attr.accesskey.qname;      %Character.datatype;      #IMPLIED"
>

<!-- Defining additional attributes to div element. -->
<!ATTLIST %div.qname;
    %BML.accesskey.attrib;
>

<!-- Defining additional attributes to p element. -->
<!ATTLIST %p.qname;
    %BML.accesskey.attrib;
>

<!-- Defining additional attributes to span element. -->
<!ATTLIST %span.qname;
    %BML.accesskey.attrib;
>

<!-- Defining additional attributes to object element. -->
<!ENTITY % BML.object.remain.qname "%BML.pfx;remain" >
<!ENTITY % BML.object.streamstatus.qname "%BM.pfx;streamstatus" >
<!ENTITY % BML.object.streamposition.qname "%BM.pfx;streamposition" >
<!ENTITY % BML.object.streamlooping.qname "%BM.pfx;streamlooping" >
<!ENTITY % BML.object.streamspeednumerator.qname "%BM.pfx;streamspeednumerator" >
<!ENTITY % BML.object.streamspeeddenominator.qname "%BM.pfx;streamspeeddenominator"
>
<!ENTITY % BML.object.streamlevel.qname "%BM.pfx;streamlevel" >
<!ATTLIST %object.qname;
    %BML.accesskey.attrib;
    %BML.presentation.remain.qname;      (remain)      #IMPLIED
    %BML.stream.streamstatus.qname;      (stop | play | pause) #IMPLIED
    %BML.stream.streamposition.qname;      %Number.datatype;      "0"
    %BML.stream.streamlooping.qname;      %Number.datatype;      "1"
    %BML.object.streamspeednumerator.qname; %Number.datatype;      "1"
    %BML.object.streamspeeddenominator.qname; %Number.datatype;      "1"
    %BML.stream.streamlevel.qname;      %Number.datatype;      "100"
>

<!-- Defining additional attributes to bdo element. -->
<![%XHTML.bidi;[
<!ENTITY % BML.bdo.orientation "%BML.pfx;orientation" >
<!ATTLIST %bdo.qname;
    %BML.bdo.orientation;      (horiz | vert)      "horiz"
>
]]> <!-- XHTML.bidi -->

<!-- Defining additional attributes to a element. -->

```

```

<!ENTITY % BML.a.effect.class
    "(cut | dissolve | wipe1 | wipe2 | wipe3 | wipe4 | goosewing1
    | goosewing2 | goosewing3 | goosewing4 | roll1 | roll2 | roll3
    | roll4 | slide-in1 | slide-in2 | slide-in3 | slide-in4
    | slide-out1 | slide-out2 | slide-out3 | slide-out4
    | separete-wipe1 | separate-wipe2 | separate-wipe3 | separate-wipe4
    | square-wipe1 | square-wipe2)"
>
<!ENTITY % BML.a.effect.qname "%BML.pfx;effect" >
<!ATTLIST %a.qname;
    %BML.a.effect.qname;      %BML.a.effect.class;      "cut"
>

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!-- defining additional events attributes entities -->
<!ENTITY % BML.events.onfocus.qname "%BML.pfx;onfocus" >
<!ENTITY % BML.events.onblur.qname "%BML.pfx;onblur" >

<!-- additional events attributes on div element
-->
<!ATTLIST %div.qname;
    %BML.events.onfocus.qname;      %Script.datatype;      #IMPLIED
    %BML.events.onblur.qname;        %Script.datatype;      #IMPLIED
>

<!-- additional events attributes on p element
-->
<!ATTLIST %p.qname;
    %BML.events.onfocus.qname;      %Script.datatype;      #IMPLIED
    %BML.events.onblur.qname;        %Script.datatype;      #IMPLIED
>

<!-- additional events attributes on span element
-->
<!ATTLIST %span.qname;
    %BML.events.onfocus.qname;      %Script.datatype;      #IMPLIED
    %BML.events.onblur.qname;        %Script.datatype;      #IMPLIED
>

<!-- additional events attributes on object element
-->
<!ATTLIST %object.qname;
    %BML.events.onfocus.qname;      %Script.datatype;      #IMPLIED
    %BML.events.onblur.qname;        %Script.datatype;      #IMPLIED
>

<!-- end of bml-elements_2_0.mod -->

```

D.3 Basic BML Extension Elements Module DTD

```

<!-- ..... -->

```

```
<!-- Basic BML Extension Elements
Module ..... -->
<!-- file: bml-basic-elements_2_0.mod

This is BML, a broadcast markup language as a modular XML application.
Copyright 2001 ARIB, All Rights Reserved.
Revision: $Id: bml-basic-elements_2_0.mod,v 1.0 2001/10/09 21:00 ARIB Data
Broadcasting WG PE-TG

This DTD module is identified by the PUBLIC and SYSTEM identifiers:

PUBLIC "-//ARIB//ELEMENTS XHTML Basic BML Elements 2.0//JA"
SYSTEM "http://www.arib.or.jp/B24/DTD/bml-basic-elements_2_0.mod"

..... -->

<!-- BML Extension Elements Module

bevent
beitem

This module defines BML elements and their attributes provided extensionnal
facilities to XHTML for broadcasting service.

And this module defines additional attributes for XHTML defined elements.
-->

<!-- ..... -->

<!-- bevent: BML Interruption Event Description Element -->

<!ENTITY % BML.bevent.content "(%BML.beitem.qname;)+ " >
<!ELEMENT %BML.bevent.qname; %BML.bevent.content; >
<!ATTLIST %BML.bevent.qname;
    %id.attrib;
>

<!-- beitem: BML Interruption Event Item Element -->

<!ENTITY % BML.beitemType.class
    "( EventMessageFired | EventFinished | EventEndNotice | Abort
    | ModuleUpdated | ModuleLocked | TransmissionFinished
    | TimerFired | DataEventChanged | CCStatusChanged
    | MainAudioStreamChanged | NPTReferred | MediaStarted
    | MediaStopped | MediaRepeated | DataButtonPressed
    | IPConnectionTerminated | PeripheralEventOccured )"
>
<!ENTITY % BML.beitemTimeMode.class
    "( absolute | origAbsolute | relativeToEvent
    | relativeToLoad | NPT)"
>
<!ENTITY % BML.beitem.content "EMPTY" >
<!ELEMENT %BML.beitem.qname; %BML.beitem.content; >
<!ATTLIST %BML.beitem.qname;
```

```

        id                ID                #REQUIRED
        type              %BML.beitemType.class;    #REQUIRED
        onoccur            %Script.datatype;        #REQUIRED
        es_ref             %URI.datatype;          #IMPLIED
        message_id        %Number.datatype;        #IMPLIED
        message_version    %Number.datatype;        #IMPLIED
        message_group_id  %Number.datatype;        #IMPLIED
        module_ref        %URI.datatype;          #IMPLIED
        language_tag      %Number.datatype;        #IMPLIED
        peripheral_ref     %URI.datatype;          #IMPLIED
        time_mode         %BML.beitemTimeMode.class; #IMPLIED
        time_value        CDATA                  #IMPLIED
        object_id         IDREF                  #IMPLIED
        subscribe         (subscribe)            #IMPLIED
    >

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!-- Defining additional attributes to body element. -->
<!ENTITY % BML.body.invisible.qname "%BML.pfx;invisible" >
<!ATTLIST %body.qname;
    %BML.body.invisible.qname;    (invisible)    #IMPLIED
>

<!-- Declaring additional attribute for div, p, span, object element -->
<!ENTITY % BML.attr.accesskey.qname "%BML.pfx;accesskey" >
<!ENTITY % BML.accesskey.attrib
    "%BML.attr.accesskey.qname;    %Character.datatype;    #IMPLIED"
>

<!-- Defining additional attributes to div element. -->
<!ATTLIST %div.qname;
    %BML.accesskey.attrib;
>

<!-- Defining additional attributes to p element. -->
<!ATTLIST %p.qname;
    %BML.accesskey.attrib;
>

<!-- Defining additional attributes to span element. -->
<!ATTLIST %span.qname;
    %BML.accesskey.attrib;
>

<!-- Defining additional attributes to object element. -->
<!ENTITY % BML.object.remain.qname "%BML.pfx;remain" >
<!ENTITY % BML.object.streamstatus.qname "%BML.pfx;streamstatus" >
<!ENTITY % BML.object.streamposition.qname "%BML.pfx;streamposition" >
<!ENTITY % BML.object.streamlooping.qname "%BML.pfx;streamlooping" >
<!ATTLIST %object.qname;
    %BML.accesskey.attrib;

```

D.4 BML Document Model Module

This is BML, a broadcast markup language as a modular XML application.
Copyright 2001 ARIB, All Rights Reserved.
Revision: \$Id: bml-model_2_0.mod, v 1.0 2001/10/09 21:03 ARIB Data
Broadcasting WG PE-TG

```
This DTD module is identified by the PUBLIC and SYSTEM identifiers:

PUBLIC "-//ARIB//ELEMENTS BML DOCUMENT MODEL 2.0//JA"
SYSTEM "http://www.arib.or.jp/B24/DTD/bml-model_2_0.mod"

..... -->

<!-- BML Document Model

This module describes the groupings of elements that make up
common content models for XHTML elements.

-->

<!-- Extending the Model

While in some cases this module may need to be rewritten to
accommodate changes to the document model, minor extensions
may be accomplished by redeclaring any of the three *.extra;
parameter entities to contain extension element types as follows:

    %Misc.extra;    whose parent may be any block or
                    inline element.

    %Inline.extra;  whose parent may be any inline element.

    %Block.extra;   whose parent may be any block element.

If used, these parameter entities must be an OR-separated
list beginning with an OR separator ("|"), eg., "| a | b | c"

All block and inline *.class parameter entities not part
of the *struct.class classes begin with "|" to allow for
exclusion from mixes.

-->

<!-- Additional Qualified Names ..... -->
<!-- xhtml-frames-1.mod -->
<!ENTITY % frameset.qname "%XHTML.pfx;frameset" >
<!ENTITY % frame.qname "%XHTML.pfx;frame" >
<!ENTITY % noframes.qname "%XHTML.pfx;noframes" >
<!-- xhtml-iframe-1.mod -->
<!ENTITY % iframe.qname "%XHTML.pfx;iframe" >

<!-- ..... Optional Elements in head ..... -->

<!ENTITY % HeadOpts.mix
    "( %script.qname; | %style.qname; | %meta.qname;
      | %link.qname; | %object.qname; | %BML.bevent.qname; )"
>

<!-- ..... Miscellaneous Elements ..... -->
```

```
<!-- ins and del are used to denote editing changes
-->
<!ENTITY % Edit.class "| %ins.qname; | %del.qname;" >

<!-- script and noscript are used to contain scripts
      and alternative content
-->
<!ENTITY % Script.class "| %script.qname; | %noscript.qname;" >

<!ENTITY % Misc.extra "" >

<!-- These elements are neither block nor inline, and can
      essentially be used anywhere in the document body.
-->
<!ENTITY % Misc.class
      "%Edit.class;
      %Script.class;
      %Misc.extra;"
>

<!-- ..... Inline Elements ..... -->

<!ENTITY % InlStruct.class "%br.qname; | %span.qname;" >

<!ENTITY % InlPhras.class
      "| %em.qname; | %strong.qname; | %dfn.qname; | %code.qname;
      | %samp.qname; | %kbd.qname; | %var.qname; | %cite.qname;
      | %abbr.qname; | %acronym.qname; | %q.qname;" >

<!ENTITY % InlPres.class
      "| %tt.qname; | %i.qname; | %b.qname; | %big.qname;
      | %small.qname; | %sub.qname; | %sup.qname; | %iframe.qname;" >

<!ENTITY % I18n.class "| %bdo.qname;" >

<!ENTITY % Anchor.class "| %a.qname;" >

<!ENTITY % InlSpecial.class
      "| %img.qname; | %map.qname;
      | %object.qname;" >

<!ENTITY % InlForm.class
      "| %input.qname; | %select.qname; | %textarea.qname;
      | %label.qname; | %button.qname;" >

<!ENTITY % Inline.extra "" >

<!-- %Inline.class; includes all inline elements,
      used as a component in mixes
-->
<!ENTITY % Inline.class
      "%InlStruct.class;
```



```
%InlPhras.class;
%InlPres.class;
%I18n.class;
%Anchor.class;
%InlSpecial.class;
%InlForm.class;
%Inline.extra;"
>

<!-- %InlNoAnchor.class; includes all non-anchor inlines,
      used as a component in mixes
-->
<!ENTITY % InlNoAnchor.class
      "%InlStruct.class;
      %InlPhras.class;
      %InlPres.class;
      %I18n.class;
      %InlSpecial.class;
      %InlForm.class;
      %Inline.extra;"
>

<!-- %InlNoAnchor.mix; includes all non-anchor inlines
-->
<!ENTITY % InlNoAnchor.mix
      "%InlNoAnchor.class;
      %Misc.class;"
>

<!-- %Inline.mix; includes all inline elements, including %Misc.class;
-->
<!ENTITY % Inline.mix
      "%Inline.class;
      %Misc.class;"
>

<!-- ..... Block Elements ..... -->

<!-- In the HTML 4.0 DTD, heading and list elements were included
      in the %block; parameter entity. The %Heading.class; and
      %List.class; parameter entities must now be included explicitly
      on element declarations where desired.
-->

<!ENTITY % Heading.class
      "%h1.qname; | %h2.qname; | %h3.qname;
      | %h4.qname; | %h5.qname; | %h6.qname;" >

<!ENTITY % List.class "%ul.qname; | %ol.qname; | %dl.qname;" >

<!ENTITY % Table.class "| %table.qname;" >
```

```
<!ENTITY % Form.class  "| %form.qname;" >

<!ENTITY % Fieldset.class  "| %fieldset.qname;" >

<!ENTITY % BlkStruct.class "%p.qname; | %div.qname;" >

<!ENTITY % BlkPhras.class
    "| %pre.qname; | %blockquote.qname; | %address.qname;" >

<!ENTITY % BlkPres.class  "| %hr.qname;" >

<!ENTITY % BlkSpecial.class
    "%Table.class;
    %Form.class;
    %Fieldset.class;
    | %noframes.qname;"
>

<!ENTITY % Block.extra "" >

<!-- %Block.class; includes all block elements,
    used as an component in mixes
-->
<!ENTITY % Block.class
    "%BlkStruct.class;
    %BlkPhras.class;
    %BlkPres.class;
    %BlkSpecial.class;
    %Block.extra;"
>

<!-- %Block.mix; includes all block elements plus %Misc.class;
-->
<!ENTITY % Block.mix
    "%Heading.class;
    | %List.class;
    | %Block.class;
    %Misc.class;"
>

<!-- ..... All Content Elements ..... -->

<!-- %Flow.mix; includes all text content, block and inline
-->
<!ENTITY % Flow.mix
    "%Heading.class;
    | %List.class;
    | %Block.class;
    | %Inline.class;
    %Misc.class;"
>

<!-- redeclare content model of <html> to allow for either
```

body or frameset content. The SGML markup minimization features used in HTML 4 do not apply, so the ambiguity that necessitated separation into the separate Frameset and Transitional DTDs is eliminated.

```
-->
<!ENTITY % html.content
    "( %head.qname;; ( %body.qname; | %frameset.qname; ) )"
>
<!-- end of bml-model_2_0.mod -->
```

D.5 BML qname Module

```
<!-- ..... -->
<!-- BML Qname Module ..... -->
<!-- file: bml-qname_2_0.mod
    This is BML, a broadcast markup language as a modular XML application.
    Copyright 2001 ARIB, All Rights Reserved.
    Revision: $Id: bml-qname_2_0.mod, v 1.0 2001/10/09 21:06 ARIB Data
        Broadcasting WG PE-TG

    This DTD module is identified by the PUBLIC and SYSTEM identifiers:

        PUBLIC "-//ARIB//ENTITIES BML Qualified Names 2.0//JA"
        SYSTEM "http://www.arib.or.jp/B24/DTD/bml-qname_2_0.mod"

    ..... -->

<!-- Bring in the datatypes - we use the URI.datatype PE for declaring the
    xmlns attributes. -->
<!ENTITY % XHTML-datatypes.mod
    PUBLIC "-//W3C//ENTITIES XHTML Datatypes 1.0//EN"
        "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-datatypes-1.mod" >
%XHTML-datatypes.mod;

<!-- By default, disable prefixing of this module -->
<!ENTITY % NS_BML.prefix "INCLUDE" >
<!ENTITY % BML.prefix "%NS_BML.prefix;" >

<!-- Declare the actual namespace of this module -->
<!ENTITY % BML.xmlns "http://www.arib.or.jp/B24/DTD/bml" >

<!-- Declare the default prefix for this module -->
<!ENTITY % BML.prefix "bml" >

<!-- If this module's namespace is prefixed -->
<![%BML.prefix;[
    <!ENTITY % BML.pfx "%BML.prefix;" >
]]>
<!ENTITY % BML.pfx "" >

<!-- Declare a Parameter Entity (PE) that defines any external namespaces
    that are used by this module -->
<!ENTITY % BML.xmlns.extra.attrib "" >
```

```
<!-- Declare a PE that defines the xmlns attributes for use by BML. -->
<![%BML.prefixed;[
<!ENTITY % BML.xmlns.attrib
    "xmlns:%BML.prefix; %URI.datatype; #FIXED '%BML.xmlns;'
    %BML.xmlns.extra.attrib;"
>
]]>
<!ENTITY % BML.xmlns.attrib
    "xmlns %URI.datatype; #FIXED '%BML.xmlns;'
    %BML.xmlns.extra.attrib;"
>

<!-- Make sure that the BML namespace attributes are included on the XHTML
    attribute set -->
<![%NS_BML.prefixed ;[
<!ENTITY % XHTML.xmlns.extra.attrib
    "%BML.xmlns.attrib;" >
]]>
<!ENTITY % XHTML.xmlns.extra.attrib
    ""
>

<!-- Now declare the element names -->
<!ENTITY % BML.bml.qname "%BML.pfx;bml" >
<!ENTITY % BML.bevent.qname "%BML.pfx;bevent" >
<!ENTITY % BML.beitem.qname "%BML.pfx;beitem" >

<!-- end of bml-qname_2_0.mod -->
```

D.6 Basic Mobile BML Extension Elements Module DTD

```
<!-- ..... -->
<!-- Basic Mobile BML Extension Elements
Module ..... -->
<!-- file: bml-basic-mobile-elements_2_0.mod

    This is BML, a broadcast markup language as a modular XML application.
    Copyright 2005 ARIB, All Rights Reserved.
    Revision: $Id: bml-basic-mobile-elements_2_0.mod,v 1.0 2005/05/31 21:00 ARIB
Data
    Broadcasting WG PE-TG

    This DTD module is identified by the PUBLIC and SYSTEM identifiers:

        PUBLIC "-//ARIB//ELEMENTS XHTML Basic Mobile BML Elements 2.0//JA"
        SYSTEM "http://www.arib.or.jp/B24/DTD/bml-basic-mobile-elements_2_0.mod"

    ..... -->

<!-- BML Extension Elements Module

    bevent
    beitem

    This module defines BML elements and their attributes provided extensionnal
    facilities to XHTML for broadcasting service.
```

```

    And this module defines additional attributes for XHTML defined elements.
-->

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!-- bevent: BML Interruption Event Description Element -->

<!ENTITY % BML.bevent.content "(%BML.beitem.qname;)+>
<!ELEMENT %BML.bevent.qname; %BML.bevent.content; >
<!ATTLIST %BML.bevent.qname;
    %id.attrib;
>

<!-- beitem: BML Interruption Event Item Element -->

<!ENTITY % BML.beitemType.class
    "( EventMessageFired | ModuleUpdated | ModuleLocked
    | TimerFired | DataEventChanged
    | MainAudioStreamChanged | MediaStopped )"
>
<!ENTITY % BML.beitemTimeMode.class
    "( absolute | origAbsolute )"
>
<!ENTITY % BML.beitem.content "EMPTY" >
<!ELEMENT %BML.beitem.qname; %BML.beitem.content; >
<!ATTLIST %BML.beitem.qname;
    id                ID                #REQUIRED
    type              %BML.beitemType.class;    #REQUIRED
    onoccur           %Script.datatype;    #REQUIRED
    es_ref            %URI.datatype;        #IMPLIED
    message_id        %Number.datatype;      #IMPLIED
    message_version   %Number.datatype;      #IMPLIED
    message_group_id  %Number.datatype;      #IMPLIED
    module_ref        %URI.datatype;        #IMPLIED
    time_mode         %BML.beitemTimeMode.class;    #IMPLIED
    time_value        CDATA                #IMPLIED
    object_id         IDREF                #IMPLIED
    subscribe         (subscribe)         #IMPLIED
>

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!-- Declaring additional attribute for object element -->
<!ENTITY % BML.attr.accesskey.qname "%BML.pfx;accesskey" >
<!ENTITY % BML.accesskey.attrib
    "%BML.attr.accesskey.qname;    %Character.datatype;    #IMPLIED"
>

<!-- Defining additional attributes to object element. -->
<!ENTITY % BML.object.streamstatus.qname "%BML.pfx;streamstatus" >
<!ATTLIST %object.qname;
    %BML.accesskey.attrib;
    %BML.object.streamstatus.qname;          (stop | play | pause) #IMPLIED
>

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!-- defining additional events attributes entities -->
<!ENTITY % BML.events.onfocus.qname "%BML.pfx;onfocus" >
<!ENTITY % BML.events.onblur.qname "%BML.pfx;onblur" >

```

```
<!-- additional events attributes on object element
-->
<!ATTLIST %object.qname;
    %BML.events.onfocus.qname;    %Script.datatype;    #IMPLIED
    %BML.events.onblur.qname;     %Script.datatype;    #IMPLIED
>

<!-- end of bml-basic-mobile-elements_2_0.mod -->
```

D.7 Server BML Extension Elements Module DTD

```
<!-- ..... -->
<!-- Server BML Extension Elements
Module ..... -->
<!-- file: bml-server-elements_1_0.mod

    This is BML, a broadcast markup language as a modular XML application.
    Copyright 2006 ARIB, All Rights Reserved.
    Revision: $Id: bml-server-elements_1_0.mod,v 1.0 2006/03/01 0:00 ARIB Data
        Broadcasting WG PE-TG

    This DTD module is identified by the PUBLIC and SYSTEM identifiers:

        PUBLIC "-//ARIB//ELEMENTS XHTML Server BML Elements 1.0//JA"
        SYSTEM "http://www.arib.or.jp/B24/DTD/bml-server-elements_1_0.mod"

    ..... -->

<!-- BML Extension Elements Module

    bevent
    beitem

    This module defines BML elements and their attributes provided extensionnal
    facilities to XHTML for broadcasting service.

    And this module defines additional attributes for XHTML defined elements.
-->

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!-- bevent: BML Interruption Event Description Element -->

<!ENTITY % BML.bevent.content "(%BML.beitem.qname;)+>
<!ELEMENT %BML.bevent.qname; %BML.bevent.content; >
<!ATTLIST %BML.bevent.qname;
    %id.attrib;
>

<!-- beitem: BML Interruption Event Item Element -->

<!ENTITY % BML.beitemType.class
    "( EventMessageFired | ModuleUpdated | ModuleLocked
    | TimerFired | DataEventChanged | CCStatusChanged
    | MainAudioStreamChanged | NPTReferred | MediaStarted
    | MediaStopped | DataButtonPressed
    | IPConnectionTerminated | StoreFinished
    | DataEventChangedEx | SegmentPlayEnded | MetadataUpdated )"
>
<!ENTITY % BML.beitemTimeMode.class
    "( absolute | origAbsolute | NPT )"
>
<!ENTITY % BML.beitem.content "EMPTY" >
```

```
<!ELEMENT %BML.beitem.qname; %BML.beitem.content; >
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!-- Defining additional attributes to body element. -->
<!ENTITY % BML.body.invisible.qname "%BML.pfx;invisible" >
<ATTLIST %body.qname;
    %BML.body.invisible.qname;      (invisible)      #IMPLIED
>

<!-- Declaring additional attribute for div, p, span, object element -->
<!ENTITY % BML.attr.accesskey.qname "%BML.pfx;accesskey" >
<!-- Defining additional attributes to div element. -->
<ATTLIST %div.qname;
    %BML.attr.accesskey.qname;      %Character.datatype;      #IMPLIED
>

<!-- Defining additional attributes to p element. -->
<ATTLIST %p.qname;
    %BML.attr.accesskey.qname;
>

<!-- Defining additional attributes to span element. -->
<ATTLIST %span.qname;
    %BML.attr.accesskey.qname;
>

<!-- Defining additional attributes to object element. -->
<!-- defining additional events attributes entities -->
```

```
<!ENTITY % BML.events.onfocus.qname "%BML.pfx;onfocus" >
<!ENTITY % BML.events.onblur.qname "%BML.pfx;onblur" >

<!-- additional events attributes on div element
-->
<!ATTLIST %div.qname;
    %BML.events.onfocus.qname;    %Script.datatype;    #IMPLIED
    %BML.events.onblur.qname;      %Script.datatype;    #IMPLIED
>

<!-- additional events attributes on p element
-->
<!ATTLIST %p.qname;
    %BML.events.onfocus.qname;    %Script.datatype;    #IMPLIED
    %BML.events.onblur.qname;      %Script.datatype;    #IMPLIED
>

<!-- additional events attributes on span element
-->
<!ATTLIST %span.qname;
    %BML.events.onfocus.qname;    %Script.datatype;    #IMPLIED
    %BML.events.onblur.qname;      %Script.datatype;    #IMPLIED
>

<!-- additional events attributes on object element
-->
<!ATTLIST %object.qname;
    %BML.events.onfocus.qname;    %Script.datatype;    #IMPLIED
    %BML.events.onblur.qname;      %Script.datatype;    #IMPLIED
>

<!-- end of bml-server-elements_1_0.mod -->
```


Annex E Resource List for Content to Be Received in Real Time

This section defines a resource list describing resource information that defines a namespace, which is independent of any transmission allowing a content to be received in real time.

To use a set of resources to be used by a real-time content in the real-time location method for a data broadcasting service, this resource list defined in this section is transmitted. The media type (Content-type) of the resource list must be "application/X-arib-realTimeLocationResourceList". The real-time location resource list is defined in Table E-1.

Table E-1 Encoding of Real-time Location Resource List

Syntax	Number of bits	Mnemonic
X-arib-realTimeLocationResourceList{		
realTimeLocationResourceListLength	32	uimsbf
realTimeLocationRootNameLength	8	uimsbf
for (i=0; i< realTimeLocationRootNameLength; i++) {		
text_char	8	uimsbf
}		
num_of_files	16	uimsbf
for(j=0;i< num_of_files;j++){		
directory_flag	1	bslbf
reserve	7	bslbf
if (directory_flag == "1"){		
directoryInfo()		
} else {		
fileInfo()		
}		
}		
}		

Semantics of X-arib-realTimeLocationResourceList()

realTimeLocationResourceListLength (Resource List Length):

This 32-bit field indicates the byte length of the area from the beginning of the immediately following field to the end of the resource list.

realTimeLocationRootNameLength (Root Directory Name Length):

This 7-bit field indicates the length in bytes of the following root directory name.

text_char (Root Directory Name):

This 8-bit field contains a string representing the name of the root directory on the resource list. The string is up to 127 bytes. The root directory name is referred to as realTimeLocationRootName.

num_of_files (Number of Files):

This field indicates the number of the directories and resources (files) immediately under realTimeLocationRootName in this namespace.

directoryFlag (directory flag): This 1-bit field indicates whether an element located immediately under the directory is a file or a directory.

Value	Semantics
0	The element is a file (resource).
1	The element is a directory.

directoryInfo() (directory information):

This field contains information about the directory, as defined in Table E-2.

fileInfo() (file information):

This field contains information about the file, as defined in Table E-3.

Table E-2 Encoding of directoryInfo()

Syntax	Number of bits	Mnemonic
directoryInfo() {		
DirectoryInfoLength	32	uimbsf
DirectoryNameLength	8	uimbsf
for (j=0; j< directoryNameLength; j++) {		
text_char	8	uimbsf
}		
num_of_files	16	uimbsf
for (i=0; i< num_of_files; i++) {		
DirectoryFlag	1	bslbf
Reserved	7	bslbf
if (directoryFlag == "1") {		
directoryInfo()		
} else {		
fileInfo()		
}		
}		

Semantics of directoryInfo()

directoryInfoLength (Directory Information Length):

This 32-bit field indicates the byte length of the area from the beginning of the immediately following field to the end of the concerned directory information.

directoryNameLength (directory name length):

This 7-bit field indicates the byte length of the following directory name.

text_char (directory name):

This 8-bit field contains a string representing a directory name. The string is up to 127 bytes.

num_of_files (number of files):

The number of directories and files located under the directory.

directoryFlag (directory flag):

This 1-bit field indicates whether an element located immediately under the concerned directory is a file or a directory.

Value	Semantics
0	The element is a file (resource).
1	The element is a directory.

directoryInfo() (directory information):

This field contains information about a directory located immediately under the directory. This implies that a certain piece of directory information may be recursively used according to a directory structure.

fileInfo() (file information):

This field contains information about the file, as defined in Table E-3.

Table E-3 Encoding of fileInfo()

Syntax	Number of bits	Mnemonic
fileInfo(){		
FileInfoLength	16	uimsbf
ResourceOffset	32	uimsbf
HeaderLength	16	uimsbf
ResourceLength	32	uimsbf
resourceTypeValue()	16	bslbf
FileNameLength	8	uimsbf
for (j=0; j< fileNameLength; j++) {		
text_char	8	uimsbf
}		
AdditionalFileInfoLength	16	uimsbf
for (j=0 ; j< N; j++) {		
AdditionalFileInfo	8	uimsbf
}		
}		

Semantics of fileInfo ():

fileInfoLength (File Information Length):

This 16-bit field indicates the byte length of the area from the beginning of the immediately following field to the end of the resource information.

resourceOffset (Resource Offset):

This 32-bit field indicates the offset in bytes of the beginning of body-part of the resource from the beginning of the module. When a single resource is mapped to a single module, this field must contain "0".

header_length:

This 16-bit field indicates the byte length of the header area in body-part of the resource specified by this resource information. This length does not include the length of CRLF (two bytes) which is inserted as a delimiter before entity-body. When a single resource is mapped to a single module, this field must contain "0".

resourceLength:

This 32-bit field indicates the length of the resource specified by this resource information. This value must be equivalent to the value of Content-Length field in the header area of body-part. When a single resource is mapped to a single module, no Content-Length field exists. Thus, this field must contain "0".

resourceTypeValue() (Resource Media Type):

This is a 16-bit data structure and indicates the media type of the resource specified by the resource information. The detailed data structure is defined in Table 9-3, B-24.

fileNameLength (File Name Length):

This 7-bit field indicates the length in bytes of the following file name.

text_char (File Name):

This 8-bit field contains a string representing a file name. The string is up to 127 bytes. The file name specified with this field must be equivalent to the resource name stored in Content-Location in the header area of body-part. When resource names are separated with "/", the file name is equivalent to a string preceded by the last "/". When a single resource is mapped to a single module, no Content-Length field exists. Thus, a file name that is not equivalent to the resource name is accepted.

additionalFileInfoLength (additional file information length):

This 16-bit field indicates the number of bytes of the following private file data area.

additionalFileInfo (additional file information):

This 8-bit field contains descriptors and values, as required, described in Table E-4, which is part of the data structure of the descriptors defined in Section 6.2.3, Volume 3. This table contains an additional descriptor, TransportLocation descriptor, of which tag value is "0xEF".

Table E-4 Available Values and Descriptors to additionalFileInfo

Tag value	Descriptor	Semantics
0x01	Type descriptor	Type of a module/file
0x02	Reserved	
0x03	Info descriptor	Character type of a module/directory/file
0x04	Reserved	
0x05	Reserved	
0x06	Reserved	
0x07	Reserved	
0x08	Reserved	
0x09 ~ 0x7F	Reserved for future use	
0x80 ~ 0xBF	The available tag values to a descriptor defined by a broadcaster	
0xC0	Expire descriptor	Expiration date for a module/directory/file
0xC1	ActivationTime descriptor	Activation time of a module/directory/file
0xC2	Compression Type descriptor	Used compression algorithm, if any
0xC3	Control descriptor	Information required to control/interpret a module/directory/file
0xC4	ProviderPrivate descriptor	Used to add proprietary information by a broadcaster or other provider
0xC5	Reserved	
0xC6	Reserved	
0xC7	Title descriptor	Used to indicate a title to describe a module/directory/file in a list to be presented to end users
0xC8	DataEncoding descriptor	Used to identify the data coding specification for proprietary data in a module/resource
0xC9	Time-stamped TS descriptor	Used to add information for transmission of MPEG video/audio in a time-stamped TS format defined in Section 8.1.4.3, Volume 2, in a data carousel
0xCA	Root certificate descriptor	Used to identify a root certificate, which is contained in a module and is applied to an interaction channel telecommunication process
0xCB	Encrypt descriptor	See Part 2, ARIB STD-B25
0xCC	ACG descriptor	See Part 2, ARIB STD-B25
0xCD ~ 0xED	Reserved for future use	
0xEE	MetadataFragment descriptor	Used to identify the ID and version of metadata contained in a module
0xEF	TransportLocation descriptor	See Table E-5
0xF0~ 0xFF	Reserved for descriptor tags identifying a data coding specification	

Table E-5 TransportLocation descriptor

Syntax	Number of bits	Mnemonic
TransportLocation_descriptor(){ descriptor_tag descriptor_length location_type Reserved if (location_type == "000"){ module_id } else if (location_type == "001"){ component_tag module_id } else{ for (i=0;i<N;i++){ reserved } } }	8 8 3 5 8 16 8 16 8	uimbsbf uimbsbf bslbf bslbf uimbsbf uimbsbf uimbsbf uimbsbf uimbsbf

Semantics of TransportLocation descriptor:

location_type (Location Type):

This 3-bit field indicates the type of the following location field.

Value	Semantics
000	The field contains only module id.
001	The field contains component_tag and module_id.
010-111	Reserved for future use

module_id (Module Identification information):

This 16-bit field indicates the module identification information of a module through which the resource is transmitted.

component_tag (Component Tag):

This 8-bit field indicates the component tag of a data carousel, through which the resource is transmitted.

To use the aforementioned resource list to specify a file, the namespace defined below must be used.

- For a logical path or a name for identifying a file

To identify a file in a data carousel, a name in the following format must be used.

arib-rtl://<realTimeLocationRootName>*(/< directoryName>)/<fileName>

The valid characters to a directory (including a root directory) name and a file name are listed below.

directory name, file name = startChar*echar
echar = startChar | "-" | "."
startChar = lowalpha | upalpha | digit | "_"
lowalpha = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" |
 "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" |
 "y" | "z"

upalpha	= "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z"
digit	= "0" "1" "2" "3" "4" "5" "6" "7" "8" "9"

The listed alphabetical characters are case insensitive.

The symbol character "/" is used to identify a level of a directory in a directory hierarchical structure.

The minimum length of a directory name or a file name is one character long, while the maximum length is 16 characters long.

- Abbreviated names

The symbol character "." represents a directory to which the currently presented BML document contains. Any abbreviated name that starts with a directory name or a file name is a relative notation.

For example, in case of

`./DirE/yy.bml`

or

`DirE/yy.bml`

is specified in a BML document, which is identified with

`"arib-rtl://realTimeLocationRootName/DirA/DirB/DirC/DirD/xx.bml"`,

it is interpreted as

`"arib-rtl://realTimeLocationRootName/DirA/DirB/DirC/DirD/DirE/yy.bml"`.

The notation ".." represents a directory one level higher than the BML document.

For example, in case of

`../DirF/yy.bml`

is specified in a BML document, which is identified with

`"arib-rtl://realTimeLocationRootName/DirA/DirB/DirC/DirD/xx.bml"`,

it is interpreted as

`"arib-rtl://realTimeLocationRootName/DirA/DirB/DirC/DirF/yy.bml"`.

Any abbreviated name that starts with the symbol character "/" is a relative notation, representing a location relative to a directory specified with `<realTimeLocationRootName>`.

For example, when

`/DirC/yy.bml`

is specified in a BML document, which is identified with `"arib-rtl://realTimeLocationRootName/DirA/xx.bml"`,

it is interpreted as

`"arib-rtl://realTimeLocationRootName/DirC/yy.bml"`.

Informative Explanation

1 Relationship between B-XML Architecture and Multimedia Description Language BML, and Guarantee of Future Evolution

- The XML application language that is defined in this Standard includes only tags and attributes used for MM(Multimedia) encoding. This application language is called BML(Broadcast Markup Language). The scope of application corresponds to the scope that has been defined based on the requirements from multimedia services.
- XML tags that are specific to each application are defined by the DTD for that application. When a XML document is represented on a terminal, the XML tags are converted into BML tags by XSLT. The XML architecture defined in this way is called B-XML (Broadcast XML). Figure 1 illustrates the relationship between BML and B-XML. As the figure indicates, B-XML is an architectural extension to process the DTD of a BML document for each application. When a B-XML document is presented, it is treated as a BML document.

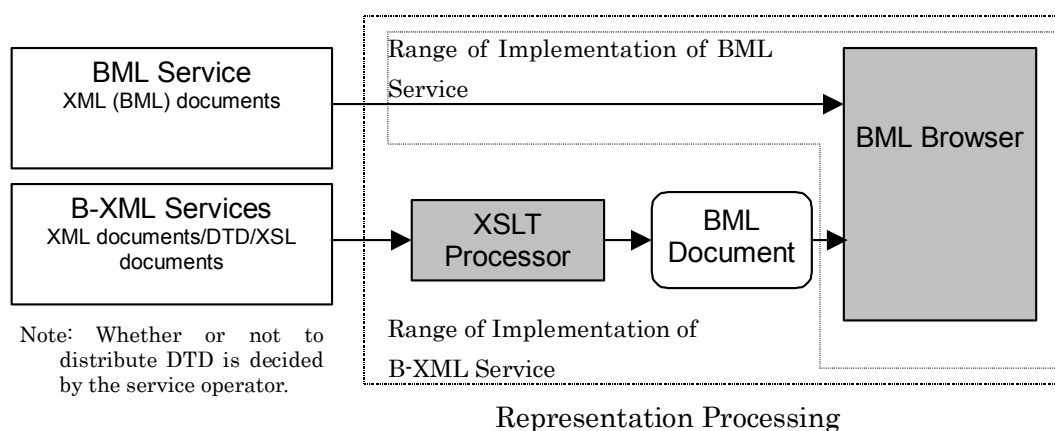


Figure 1 Implementation of XML-based Multimedia Application Language BML and B-XML Applications

- XSL is a style sheet language that defines the way to display XML documents. This Standard uses XSL as a language to convert a XML document into MM-encoded tags. Therefore it uses the specification for tag transformation (XSLT), not the formatting specification.
- There is a possibility that the DTDs and style sheets for tags and attributes that are defined by BML for MM representation might be extended to support application to other media. For this reason, a version number that consists of the major and minor numbers has been adopted to the specifications. Update of major number indicates that the processor that was designed for older versions cannot play the new version.

For optional APIs, their normal operation is ensured by providing APIs to verify that the browser has required functionality.

2 Audio Playback Control

The audio playback control follows a model shown in Figure 2.

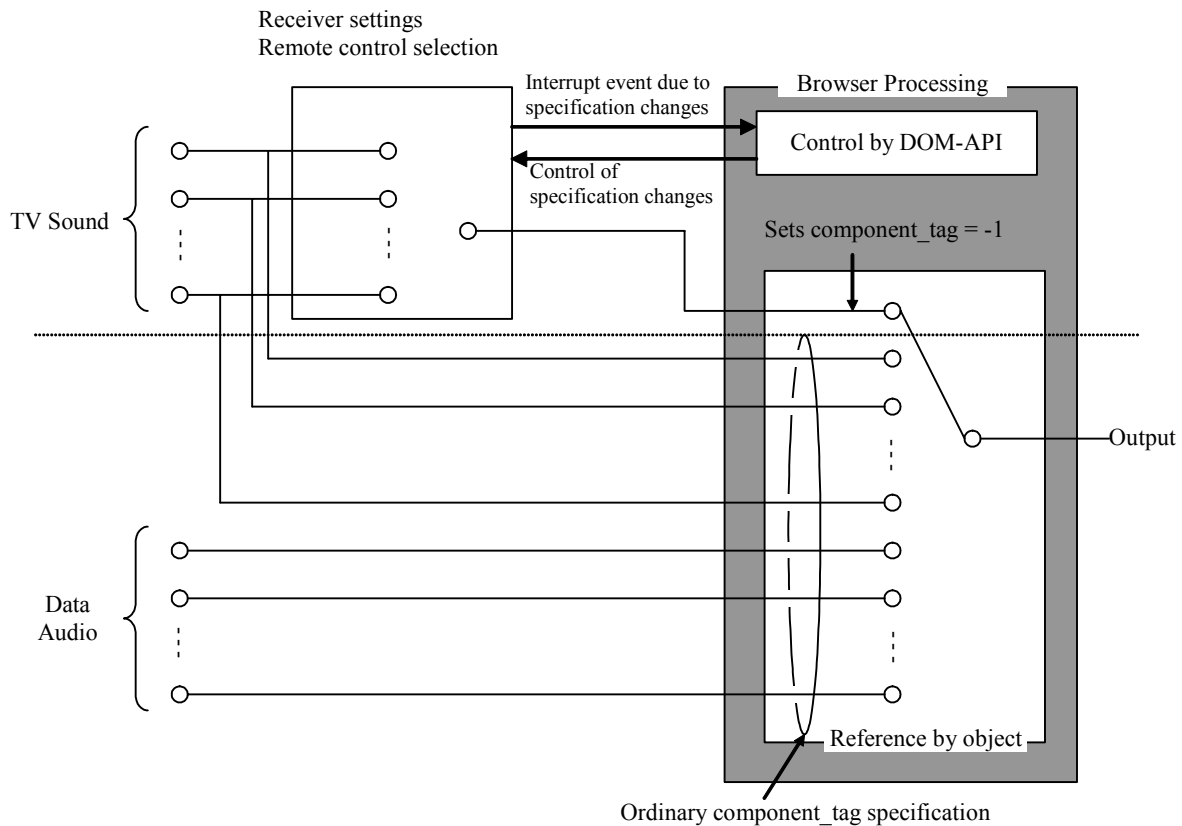


Figure 2 Reference Model of Audio Playback

- If TV audio is referenced by setting `component_tag` of object element to -1, the audio that is set or selected at the receiver is played. In this specific case, selection and setting with an audio switching DOM-API, or `selectMainAudioStream()` is supported.
- If TV audio is referenced by setting `component_tag` of object element to `component_tag` of the stream, selection and setting with above DOM-API that is cooperative with the receiver setting is not supported.
- For data audio, only reference by setting `component_tag` of object element to `component_tag` of the stream is possible. The audio switching DOM-API cannot be used.

3 Multiplexing of Still Picture Carousels and Receiver Operation

Figure 3 summarizes the multiplexing of still picture carousels that is defined in Section 8.4.3.

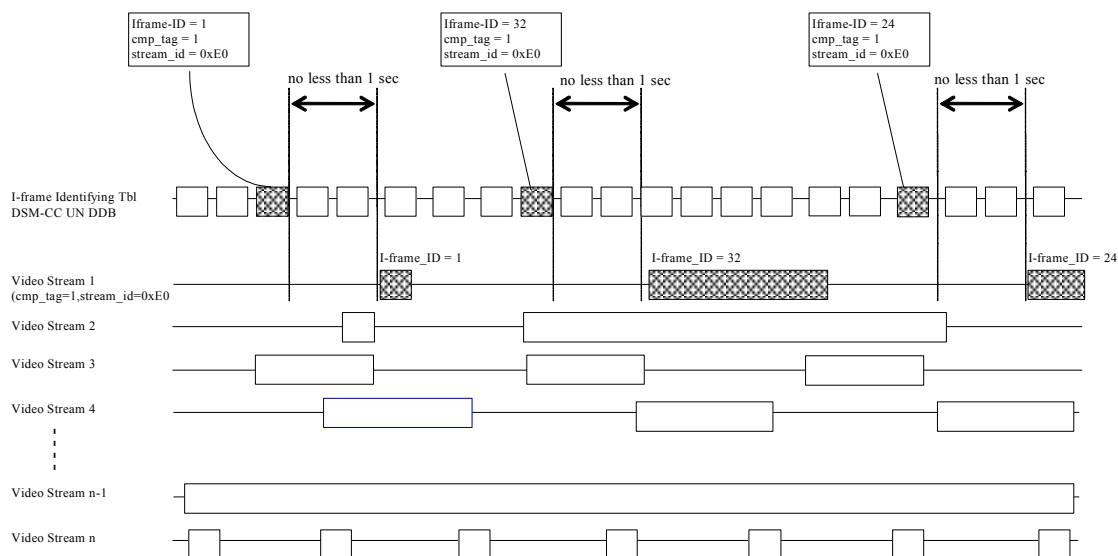


Figure 3 Multiplexing of Frames and IITs

Two or more still pictures are multiplexed on a single video ES. Section data for each still picture is multiplexed so that one still picture can be specified from the video ES. This section data is called I-frame Identifying Table (IIT). It is carried as a small module made of one DDB of a data carousel. IIT contains component_tag and stream_id that work together to identify the video ES in that I-frames are multiplexed, and information on times when display of the corresponding still picture starts and ends. The decoder decodes a requested still picture by starting and stopping decoding of video PES according to IIT information.

4 Name sharability between real-time data services and stored data services

The prefix arib-file: is designed to add a transmission-independent content identification to the existing description format.

To map a content, the combination of the following descriptions has been used to identify the content:

<original_network_id, transport_stream_id, service_id>
and
<content_id>

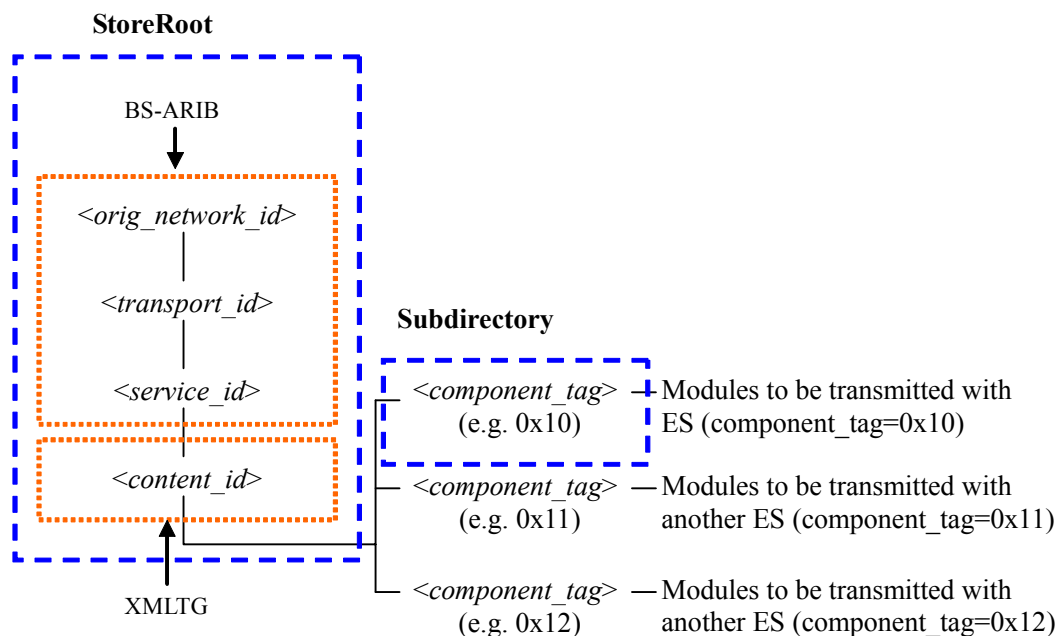
The same content is identified with the combination of the following descriptions by using the arib-file: prefix:

<rootName>
and
<subrootName>

The new namespace convention enables contents to be separately transmitted through different services to be stored as the same content. This namespace does not conflict the existing mapping system described above.

Suppose that a broadcaster (BS-ARIB) applies the following operational rules to content identification:

- StoreRoot descriptor: BS-ARIB/XMLTG
- The fixed service_id/content_id are applied to BS-ARIB /XMLTG.
- <component_tag> is applied to Subdirectory descriptor.



A file in the content is referenced as the following:

arib-file:// BS_ARIB/XMLTG/<component_tag>/<moduleName>
[/< resourceName>]

The directory structure immediately below component tag is equivalent to the structure described with the existing mapping system. This means that using the following relative specification in the content enables the content to retain the intended reference configuration to be used for several services without any modification:

```
    /<component_tag>/<moduleName>/<resourceName>  
    <moduleName>/<resourceName>  
    <resourceName>
```

With the relative specification of the namespace described above, a content which has been produced with the existing authoring tool is also applied to stored data services. A single content is able to be shared among various services.

However, to ensure the sharability of a content, the following constraints are required:

- The character "/" must not be used for resourceName because different semantics of "/" exist for resourceName.
- Any abbreviated name with "~" must not be used.
- The available characters to resourceName are governed by the namespace conventions defined in Section 9.2.15.2.

5 Sample of controlling external device by using External XML document

To control an external device by using an External XML document, it is required to have a controlling BML document specify a formal public identifier of DTD in the argument of the enumPeripherals() function to obtain a list of URIs of the external devices that are ready to process the specified DTD. This, in turn, requires each external device to be associated with the specific DTD.

An External XML document to control an external device can be obtained by either of the following methods:

- (1) The method in which a controlling External XML document is generated under the control of an BML document. First, a DOM tree is generated from an External XML document by using an XML document object. Then, the XML document object is manipulated as a Document object with DOM interface. Finally, an External XML document applicable to the external device is generated from the XML document object.
- (2) The method in which a received External XML document is passed directly to an external device. An External XML document received by using the passXMLDocToPeripheral() extended function is passed directly to an external device.

A sample process of controlling an external device is described below.

- Sample of passing data to external device

- [1] The URIs of the external devices that are connected to be ready for exchanging data by using an XML document are obtained by using the enumPeripherals() function. In this case, by specifying a formal public identifier as an argument for enumPeripherals(), the URIs of the external devices that support the specified DTD are to be obtained.
- [2] An external device to be communicated is selected from the enumerated external devices and its URI is retained.
- [3] To read an External XML document transmitted in a data carousel into a DOM tree, an XMLDoc object is generated. An External XML document is read in by using the read() method. For the explanation purpose, this sample assumes that a template of a command applicable to an external device. However, data presented directly by a BML content is also applicable.
- [4] A Document object is obtained from the XMLDoc object by using the getDocument() method.
- [5] Required information including parameters for the external device are configured in the Document object by using the DOM interface.
- [6] The modified XMLDoc object is transmitted to the external device as an External XML document by using the write() method that uses the external device's URI obtained in [1] as an argument.

- Sample of receiving data from external device

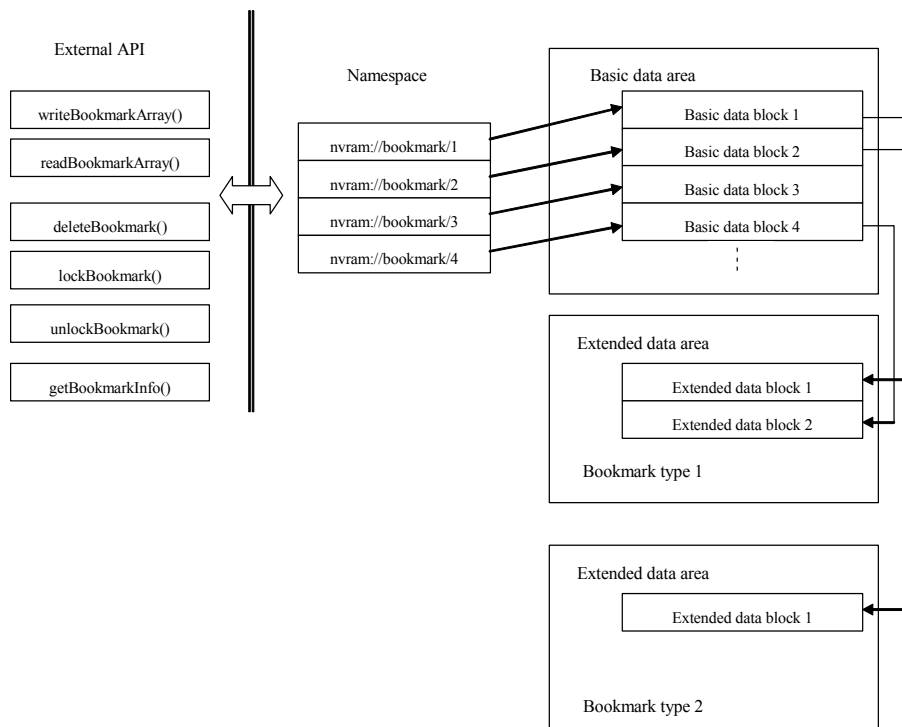
- [7] To identify a pass request of the External XML document from the external device on the content side, an event is configured. Receiving an External XML document transmitted in a data carousel is out of the scope of the configured event.
- [8] When data is received from the external device, the configured event occurs. The read () method of the XMLDoc object that uses the external device URI as an argument is executed.
- [9] The BML content processes presentation or others by using the data received from the external device.

6 Overview of Bookmark

This section outlines a bookmark.

- Data structure and accessing method

The data structure of bookmarks and how bookmarks are accessed are outlined below.



- Elements of Basic Data Block

The data structure of a bookmark consists of the basic data area and the extended data area. The basic data area consists of two or more basic data blocks, and a basic data block, in turn, is made up of the eight basic data elements shown in the following table. The extended data area consists of two or more extended data blocks. The available extended data elements are defined in an operational rule.

	Element	Semantics	Remarks
1	title	Title of bookmark block	
2	dstURI	Link destination URI	
3	expire	Expiration date	
4	registerDate	Registered date and time	
5	bmLock	Flag indicating whether or not any deletion operation is allowed.	
6	bmType	Bookmark type	
7	linkMedia	Media type for Link destination	
8	usageFlag	Flag indicating availability with/without using extended data area	

- Accessing method

To access bookmark data, for example, each basic data block is made accessible by identifying a basic data block with `nvrnm://bookmark/<block number>`. When an extended data block corresponding to a basic data block exists, the extended data block that corresponds to a number of the basic data block is accessible.

List of Extended APIs

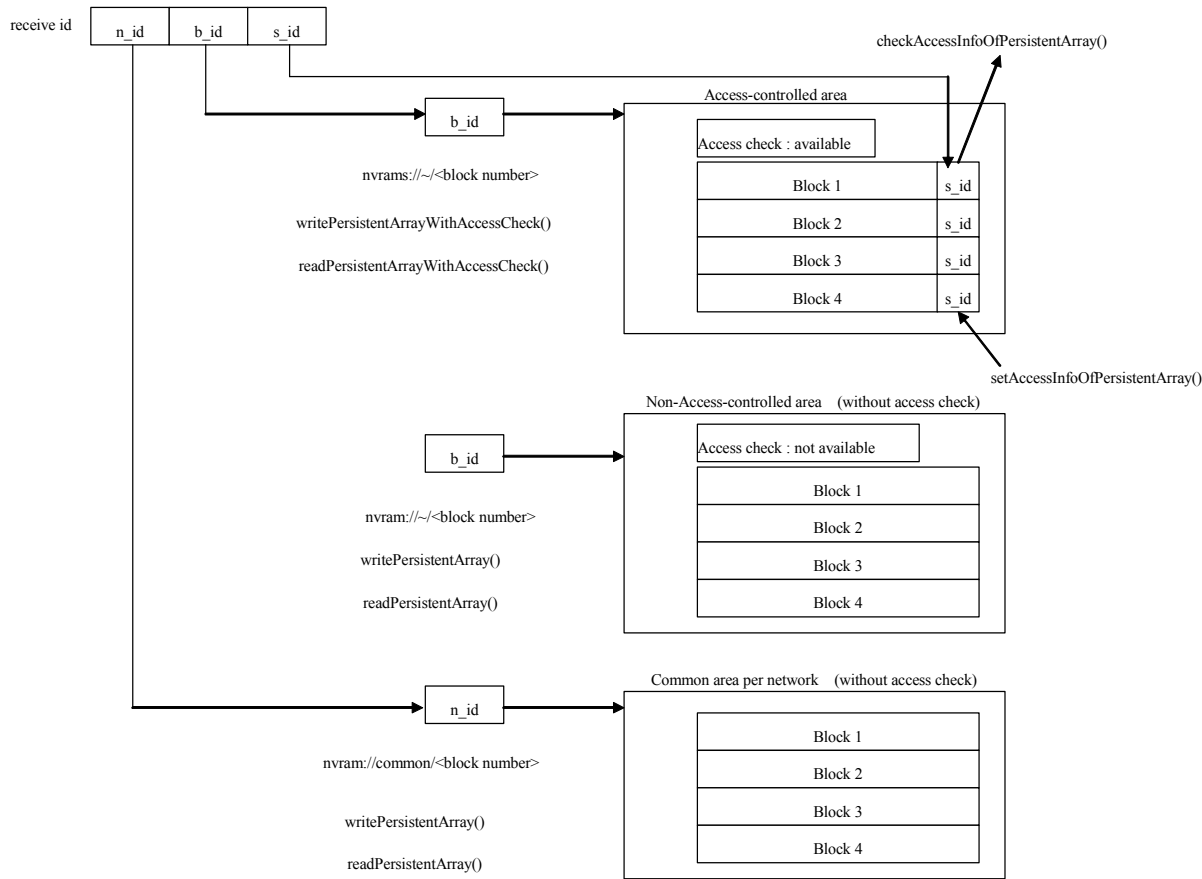
	Function	Semantics	Remarks
1	writeBookmarkArray()	Write to bookmark area	
2	readBookmarkArray()	Read out from bookmark area	
3	deleteBookmark()	Delete bookmark	
4	lockBookmark()	Makes bookmark nondeletable	
5	unlockBookmark()	Makes locked bookmark deletable	
6	getBookmarkInfo()	Obtains bookmark information	

7 Access-controlled area and non-access-controlled area in non-volatile memory

- Data structure and accessing method

Access-controlled areas and non-access-controlled areas in NVRAM are outlined in the following figure. For the brief explanation, this section applies the following description to the samples.

n_id: original_network_id
b_id: broadcaster_id
s_id: service_id



- Namespace

A new schema, `nvrms:` is introduced to access access-controlled areas.

The following table contains assumed operation per schema.

	Namespace	Schema Semantics	Sample Area
1	<code>nvrms://~/<block number></code>	access-controlled NVRAM	Area exclusive to a broadcaster *
2	<code>nvrms://~/< block number ></code>	non- access-controlled NVRAM	
3	<code>nvrms://common/< block number ></code>	non- access-controlled NVRAM	Common area for BS Common area for broadband CS broadcasters
4	<code>nvrms://bookmark/< block number ></code>	non- access-controlled NVRAM	Bookmark area

- * An actually used area, an access-controlled NVRAM (nvrams://~/<block number>) or non-access-controlled NVRAM (nvram://~/< block number >) is used, is defined in an operational rule for each media.

- Accessing method to area

List of APIs

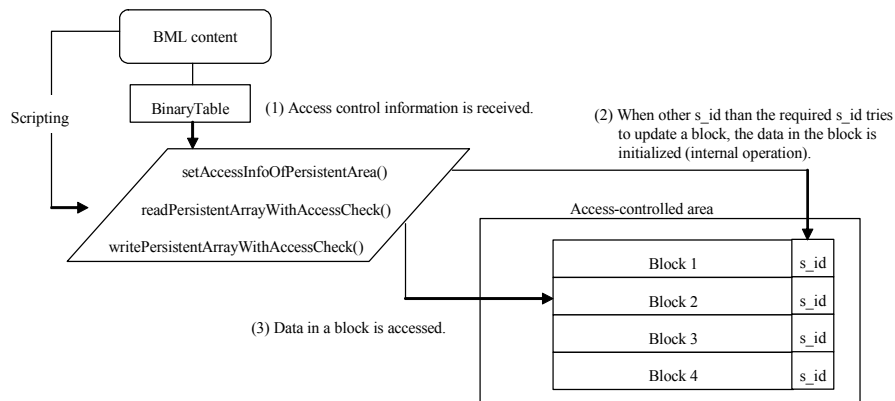
	Function	Area	Id for access check(sample)
1	readPersistentArrayWithAccessCheck()	Reads from access-controlled area	n_id b_id s_id(*1)
2	writePersistentArrayWithAccessCheck()	Writes to access-controlled area	n_id b_id s_id(*1)
3	setAccessInfoOfPersistentArray()	Configures information for access control	-
4	checkAccessInfoOfPersistentArray()	Obtains information for access control	-
5	readPersistentArray()	Reads from non-access-controlled area	n_id b_id (*2)
		Reads from common area	n_id(*2)
6	writePersistentArray()	Writes to non-access-controlled area	n_id b_id(*2)
		Writes to common area	n_id (*2)

*1: s_id is assumed to be configured by using setAccessInfoOfPersistentArray().

*2: readPersistentArray() and writePersistentArray() is not allowed to access to nvrams.

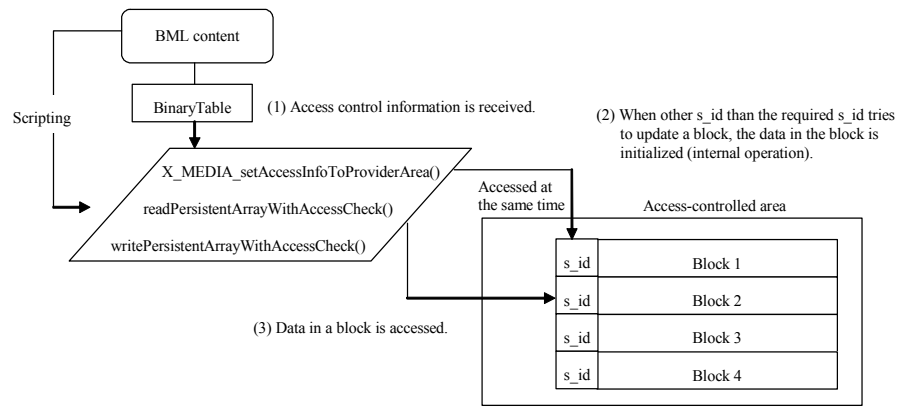
- Configuring method of access control

This section outlines access control by using setAccessInfoOfPersistentArray(). The available formats to access control information for binaryTable transmitted with contents are defined in an operational rule.



It is assumed that access-control information is written to two or more areas at the same time, as defined in an operational rule for each media.

Suppose that the responsible API is X_MEDIA_setAccessInfoToProviderArea(), access control using the API is outlines in the following figure.



References

- XML Document Structure:

- (1) Extensible Markup Language (XML) 1.0 W3C Recommendation 10-February-1998
<http://www.w3.org/TR/1998/REC-xml-19980210>
Note: The equivalent standard in Japan is JIS TR X 0008:1999, Extensible Markup Language (XML) 1.0".
- (2) Namespaces in XML World Wide Web Consortium 14-January-1999.
<http://www.w3.org/TR/1999/REC-xml-names-19990114>
- (3) JIS Standard Information (TR), TR X 0015:1999 XML, Japanese Profile
- (4) XSL Transformations (XSLT) Specification Version 1.0 W3C Working Draft 13 August 1999. <http://www.w3.org/1999/08/WD-xslt-19990813>.
- (5) Associating Style Sheets with XML documents, W3C Recommendation 29 June 1999.
<http://www.w3.org/TR/xml-styleSheet/>

- Standard Tag Set for MM Encoding

- (6) XHTML™ 1.0: The Extensible HyperText Markup Language: A Reformulation of HTML 4.0 in XML 1.0 W3C Recommendation 26 th, January, 2000
<http://www.w3.org/TR/xhtml1/>
- (7) Cascading Style Sheets, level 1 W3C Recommendation 11 January, 1999
<http://www.w3.org/TR/REC-CSS1/>
Note: The equivalent standard in Japan is JIS TR X 0011:1998 Cascading Style Sheets, level 1.
- (8) Cascading Style Sheets, level 2 CSS2 Specification W3C Recommendation 12 May, 1998
<http://www.w3.org/TR/REC-CSS2/>
- (9) W3C Recommendation "Modularization of XHTML"(10 April, 2001)
<http://www.w3.org/TR/xhtml-modularization>
- (10) WAP Forum "WAP CSS Specification Version 26-Oct-2001" (26 October, 2001)
<http://www1.wapforum.org/tech/documents/WAP-239-WCSS-20011026-a.pdf>

- Procedural Description

- (11) ECMA-262 "Standardizing Information and Communication Systems Standard ECMAScript Language Specification 2nd Edition"(August 1998)
- (12) W3C Recommendation "Document Object Model(DOM) Level 1 Specification Version 1.0"(1 October, 1998)<http://www.w3.org/TR/REC-DOM-Level-1/>
- (13) W3C Working Draft "Document Object Model (DOM) Level 2 Specification Version 1.0"(04 March, 1999)<http://www.w3.org/TR/1999/WD-DOM-Level-2-9990304>

- Others

- (14) ARIB STD B-5 "Data Multiplex Broadcasting System For the Conventional Television Using the Vertical Blanking Interval" (August, 1996)
- (15) ARIB STD B-10 V3.1 "Service Information for Digital Broadcasting System" (July, 2001)
- (16) RFC2616(June 1999)"Hypertext Transfer Protocol -- HTTP/1.1"
- (17) RFC 2068 (January 1997) "Hypertext Transfer Protocol -- HTTP/1.1"

- (18) RFC 1945(May 1996)"Hypertext Transfer Protocol -- HTTP/1.0"
- (19) RFC2046(November 1996)"Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types"
- (20) RFC1766(March 1995)"Tags for the Identification of Languages"
- (21) RFC 2965(October 2000)"HTTP State Management Mechanism"
- (22) RFC 1123(October 1989)"Requirements for Internet Hosts – Application and Support"
- (23) RFC 1808(June 1995)"Relative Uniform Resource Locators"
- (24) RFC791 (September 1981) "Internet Protocol"
- (25) RFC793 (September 1981) "Transmission Control Protocol"
- (26) RFC2131 (March 1997) "Dynamic Host Configuration Protocol"
- (27) RFC2246 (January 1999) "The TLS Protocol Version 1.0"
- (28) RFC2516 (February 1999) "A Method for Transmitting PPP Over Ethernet (PPPoE) "
- (29) JIS X0221 (1995) "Information technology – Universal Multiple-Octed Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane (ISO/IEC10646-1:1993)"
- (30) ISO/IEC 13818-1 (2000) "Information Technology – Generic Coding of Moving Pictures and Associated Audio: SYSTEMS (Second Edition) "
- (31) ISO/IEC 13818-6 (1998) "Information Technology – Generic Coding of Moving Pictures and Associated Audio Information: Extensions for Digital Storage Media Command and Control"
- (32) JIS X 0201(1997)"7-bit and 8-bit coded character sets for information interchange (ISO/IEC 646:1991) "
- (33) JIS X 0208(1997)"7-bit and 8-bit double byte coded KANJI sets for information interchange"

DATA CODING AND TRANSMISSION SPECIFICATIONS
FOR DIGITAL BROADCASTING

ARIB STANDARD

ARIB STD-B24 VERSION 5.0-E1
VOLUME2 (1/2)
(May 29, 2006)

This Document is based on the ARIB standard of “Data Coding and Transmission Specifications for Digital Broadcasting” in Japanese edition and translated into English on December, 2006.

Published by

Association of Radio Industries and Businesses

Nittochi Bldg. 11F
1-4-1 Kasumigaseki, Chiyoda-ku, Tokyo 100-0013, Japan

TEL 81-3-5510-8590
FAX 81-3-3592-1103

Printed in Japan
All rights reserved
