ENGLISH TRANSLATION

APPLICATION EXECUTION ENGINE PLATFORM FOR DIGITAL BROAD CASTING

ARIB STANDARD

ARIB STD-B23 Version 1.1

Established: June 5, 2003 Version 1.0
Revised: February 5, 2004 Version 1.1

Association of Radio Industries and Businesses

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).

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

TOTAL CONTENTS

Foreword

The ARIB (Association of Radio Industries and Businesses) has established the "ARIB standard" for the basic technical condition of standard specifications related to each radio communication equipment using radio wave and broadcasting transmission and reception equipment, with the participation of radio communication equipment manufacturers, broadcasting equipment manufacturers, electric communication companies, broadcasting companies and other users.

"ARIB standard" is a nonofficial standard established by combining governmental technical standards established for the more effective use of frequencies and to avoid interference among users, and nonofficial optional standards established for the convenience of radio communication equipment manufacturers, broadcasting equipment manufacturers, electric communication companies, broadcasting companies and users, in order to secure appropriate quality and compatibility of radio communication equipment and broadcast equipment, etc.

In order to secure fairness and transparency in drafting steps, this standard is drafted in response to a consensus of the standardization committee, with the participation of interested parties such as radio communication equipment manufacturers, broadcasting equipment manufacturers, electric communication companies, broadcasting companies, and interested users.

At this standardization committee, "Operational standard of basic construction and identifier of service information for digital broadcasting" (ARIB STD-B2), which was the standard specification related to basic construction of service information necessary to enable users to select programs, for the implementation of digital broadcasting, was established as the standard method in Japan, in May 29, 1996. As for the practical use of this standard, a data construction detail standard of service information and guideline for actual operation is necessary in addition to basic construction, so this standard, "Service information for digital broadcasting system", is established as a new nonofficial standard combining the standards mentioned above.

This standard consists of three parts. The first part includes references to other standards related to digital broadcasting and lists of tables and descriptors used in digital broadcasting, in addition to the former standard (ARIB STD-B2). The second part specifies the basic information of service information. The third part specifies the detail data construction of extension of the service information. Guidelines of operational method of service information are attached to this standard as technical documents.

Please note that in accordance with the establishment of the new standard, the former "Operational standard of basic construction and identifier of service information for digital broadcasting" (ARIB STD-B2) (May 29, 1996) is abolished.

Service information established herein considers wide application to total broadcasting media such as CS

broadcasting, BS broadcasting and digital broadcasting on the ground, preconditioning international coordination of signal structure, flexibility of program organization in each broadcasting company, and the possibility of expansion for future broadcasting service development. From now on, addition or revision of characteristic information and signals may become necessary, depending upon future developments in these broadcasting media.

We hope that this standard will be used actively among radio communication equipment manufacturers, broadcast equipment manufacturers, electric communication companies, broadcasting companies and other users.

Notice:

This standard does not describe industrial proprietary rights mandatory to this standard. However, the owner of industrial proprietary rights is expressed as "Industrial proprietary rights related to this standard, listed in the Annex below, are possessed by the applicant shown in the list. However, execution of the rights listed in the Annex below is permitted indiscriminately, without exclusion, under appropriate conditions, to the user of this standard. If 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 those rights, it is not applicable."

Annexed table (Selection of option 2)

Patent applicant	Title of the invention	Application/Patent No.	Remarks
Motorola Inc.	A comprehensive confirmation form has been submitted with regard to ARIB STD-B23 Ver. 1.1 *1.		
Nippon Hoso Kyokai (NHK)	Data Broadcast Receiver and Data Broadcast Reception Program *2	JP Kokai 2003-51799	Japan
Philips Japan Ltd.	A comprehensive confirmation form has been submitted with regard to ARIB STD-B23 Ver. 1.1 *3.		

*1: Applicable to the revised part of ARIB STD-B23 Version 1.1 (Received: January 8, 2004).

*2: Applicable to ARIB STD-B23 Version 1.0 (Received: January 27, 2004).

*3: Applicable to the revised part of ARIB STD-B23 Version 1.1 (Received: January 29, 2004).

<Blank Page>

The background of this Standard

With regard to application execution engine platform for digital broadcasting, it was considered desirable to base the Standard on the DVB system, developed by a European consortium for establishing digital broadcast standards, which is being used currently for the cable TV platform not only in Europe but also in USA, keeping in mind international interoperability and interchangeability in future digital broadcast applications, taking into account various domestic factors in stipulating the details.

This Standard was developed as a private sector standard for application execution engine platform for use in Japan, along the lines of the aforesaid policy. The Standard comprises two parts, one concerning monomedia coding systems and another concerning application execution engine platforms. The Standard embodies a system that is based on the MHP method of DVB specifications and GEM, which is a specification for international application of the said method, with additional provision of the necessary stipulations for broadcasting in the country. Therefore, compatibility with BS and CS digital broadcasting, which are already in use in Japan, is taken into account for the monomedia coding system, as also the compatibility with network utilization and the aforesaid application execution engine platforms used in western countries, for the application execution engine platform. This Standard is basically meant to be applied across all the broadcast media, and the conditions unique to certain broadcast media determined by the transmission method and service requirements are stipulated as operational restrictions.

This standard covers digital broadcasting involving application execution engine platforms. However, it is necessary to further enhance the Standard in future by adding the necessary stipulations for other broadcast media, keeping in view the trends in international standardization and the emergence of new technologies which are not even envisaged at present. The stipulation of operational restrictions and some consideration about integration with existing digital broadcasting systems are also necessary.

Part 1

MONOMEDIA CODING SYSTEM IN APPOLICATION EXCECUTION ENGINE PLATFORMS

<Blank Page>

Part 1 Monomedia Coding System in Application Execution Engine Platforms

Contents

Chapter 1 General information

1.1 Purpose

This standard stipulates the monomedia coding method in application execution engine platforms in relation to data broadcasting carried out by digital broadcasting, which is specified as a standard method in Japan.

1.2 Scope

This standard applies to monomedia coding in application execution engine platforms in data broadcasting through digital broadcasting.

1.3 References

1.3.1 Normative Documents

This Chapter defines the points of agreement with the referred standards. In this Standard, GEM1.0 refers to the following standard.

- ETSI TS 102 819 V1.2.1 Digital Video Broadcasting (DVB);
  Globally Executable MHP (GEM) Specification 1.0.1

1.4 Terms used

1.4.1 Definition of terms

The terms used in the main text of the Standard are defined below.

Monomedia:	Independent presentation media such as video, still picture, graphics, sound, text, etc.; a media that can be presented using its own data only, without referring to other media.
I Frame:	Video frame construction from coding data, which is completed within an MPEG-2 frame; Intra frame.
Chunk:	Name of a construction that expresses a block of data in PNG coding and MNG coding.
Synthesized sound:	A type of monomedia for music playback using electronic sound, etc.
Geometric:	A function to express a graphic by combining graphic description commands that specify dots, straight lines, arcs, etc.

1.4.2 Abbreviations

The following abbreviations are used in the main text of this Standard.

AAC	Advanced Audio Coding
AIFF	Audio Interchange File Format
DAVIC	Digital Audio Visual Council
DSM-CC	Digital Storage Media - Command and Control
DVB	Digital Video Broadcasting
ETSI	European Telecommunication Standards Institute
GEM	Globally Executable MHP
GIF	Graphics Interchange Format
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
JIS	Japan Industrial Standard
JPEG	Joint Picture coding Experts Group
MHP	Multimedia Home Platform
MNG	Multiple-image Network Graphics
MPEG	Moving Picture Experts Group
MUG	MHP Umbrella Group
PCM	Pulse Code Modulation
PNG	Portable Network Graphics
UCS	Universal multi-octet coded Character Set

Chapter 2 Video coding

2.1 MPEG-1 Video

MPEG-1 video coding shall be in compliance with ISO/IEC 11172-2 and by the method described in Clause 4.1, Chapter 4, Part 2, Volume 1 of ARIB STD-B24.

2.2 MPEG-2 Video

MPEG-2 video coding shall be in compliance with ISO/IEC 13818-2 (ITU-T H.262) and by the method described in Clause 4.2, Chapter 4, Part 2, Volume 1 of ARIB STD-B24.

2.3 MPEG-4 Video

MPEG-4 video coding shall be in compliance with ISO/IEC 14496-2 and by the method described in Clause 4.3, Chapter 4, Part 2, Volume 1 of ARIB STD-B24.

2.4 H.264|MPEG-4 AVC

H.264|MPEG-4 AVC video coding shall be in compliance with ITU-T Rec. H.264| ISO/IEC 14496-10 and by the method described in Clause 4.4, Chapter 4, Part 2, Volume 1 of ARIB STD-B24.

Chapter 3 Still picture and bit map coding

3.1 MPEG I picture

3.1.1 MPEG-2 I frame

MPEG-2I frame still picture coding shall be in compliance with ISO/IEC 13818-2 and by the method described in Clause 5.1.1, Chapter 5, Part 2, Volume 1 of ARIB STD-B24.

3.1.2 MPEG-4 I-VOP

MPEG-4 I-VOP still picture coding shall be in compliance with ISO/IEC 14496-2 and by the method described in Clause 5.1.2, Chapter 5, Part 2, Volume 1 of ARIB STD-B24.

3.1.3 H.264|MPEG-4 AVC I-picture

H.264|MPEG-4 AVC I-picture coding of still pictures shall be in compliance with ITU-T Rec. H 264 ISO/IEC 14496-10 and by the method described in Clause 5.1.3, Chapter 5, Part 2, Volume 1 of ARIB STD-B24.

3.2 JPEG

Bit map coding by JPEG shall be in compliance with ISO/IEC 10918-1 and by the method described in Clause 5.2, Chapter 5, Part 2, Volume 1 of ARIB STD-B24.

3.3 PNG

Bit map coding by PNG shall be in compliance with W3C recommended PNG (Portable Network Graphics) Specification Version 1.0 (W3C Recommendation 01 October, 1996) and by the method described in Clause 5.3, Chapter 5, Part 2, Volume 1 of ARIB STD-B24 and in “15 Detailed platform profile definitions” of GEM1.0. Details of applying these two coding methods are given separately as operational specification.

3.4 MNG

The file format of animation graphics by MNG shall be in compliance with MNG Format Version 0.96-19990718 and by the method described in Clause 5.4, Chapter 5, Part 2, Volume 1 of ARIB STD-B24.

3.5 MPEG-2 Video “drips”

MPEG-2 Video “drips” is an animation graphics format that uses I frame and P frame of MPEG-2 Video. The file format for animation graphics by MPEG-2 Video “drips” shall conform to GEM1.0 “15 Detailed platform profile definitions.”

3.6 GIF

GIF is a bit map coding format specified in W3C GIF89a Specification. Bit map coding by GIF shall be by the method described in GEM1.0 “15 Detailed platform profile definitions.”

Chapter 4 Audio coding

4.1 MPEG-2 Audio

Audio coding by MPEG-2 Audio shall be in compliance with the AAC system-LC profile specified in ISO/IEC 13818-7 and by the method described in Clause 6.1, Chapter 6, Part 2, Volume 1 of ARIB STD-B24.

4.2 PCM (AIFF-C)

Audio coding file format using PCM shall conform to the AIFF-C (Audio Interchange File Format) specified in DAVIC 1.4 Specification Part 9, Annex B and in accordance with Clause 6.2, Chapter 6, Part 2, Volume 1 of ARIB STD-B24.

4.3 MPEG-4 Audio

Audio coding by MPEG-4 Audio shall be in compliance with ISO/IEC 14496-3 and by the method described in Clause 6.3, Chapter 6, Part 2, Volume 1 of ARIB STD-B24.

4.4 Coding of synthesized sound

Coding of synthesized sound shall be in compliance with ARIB STB-B5 and by the method described in Clause 6.4, Chapter 6, Part 2, and Volume 1 of ARIB STD-B24.

Chapter 5 Character encoding

5.1 JIS 8-bit character code (8bit-code)

Character encoding in 8bit-code shall be in compliance with ARIB STD-B5 and by the method described in Clause 7.1, Chapter 7, Part 2, Volume 1 of ARIB STD-B24.

5.2 Universal multiple-octet coded Character Set

Character coding in Universal multiple-octet coded Character Set (UCS) shall be in compliance with ISO/IEC 10646-1:2000 and ISO/IEC 10646-1/Amd.1:2002. However, Annex A5 shall be used only for information, and all specifications other than those related to BMP shall not be used.

5.2.1 Types of character code sets and structure of code

5.2.1.1 Types of character code sets, structure of code

The coding architecture shall be 2-octet code and shall be in compliance with ISO/IEC 10646-1:2000 and ISO/IEC 10646-1/Amd.1:2002

5.2.1.2 Supplemental character (Gaiji)

The method described in GEM 1.0 “7.4 Downloadable Fonts” shall be used.

5.2.2 Enccoding scheme

UTF-8 and UTF-16 specified in ISO/IEC 10646-1:2000 shall be used as the character encoding scheme. When transmitting data, the upper byte shall be sent first in the byte order, i.e., the transmission shall be done in the “big endian” format.

5.2.3 Control characters

Only the delete character (0x007F), carriage return and line feed (0x000D, 0x000A) and tab (0x0009) shall be used.

5.2.4 Code conversion from Shift JIS

In conversion from Shift JIS, OVER LINE (code value 0x7E) specified in JIS X 0201 shall be converted to TILDE (code value 0x007E). Conversion of double byte additional characters for broadcasting in Rows 90 to 94 shall be specified in Table E-4 of Annex E of Part 2. However, the character of Row 91, Cell 8 (code value 0xEE47) shall be assigned to 0x3012 (POSTAL MARK).

5.2.5 Code conversion from EUC-JP

In conversion from EUC-JIS, OVER LINE (code value 0x7E) specified in JIS X 0201 shall be converted to TILDE (code value 0x007E). Conversion of additional characters for broadcasting

in Rows 90 to 94 in code set 1shall be in specified in Table E-4 of Annex E of Part 2. However, the character of Row 91, Cell 8 (code value 0xFBA8) shall be assigned to 0x3012 (POSTAL MARK).

5.2.6 Conversion from JIS 8-bit character code

The conversion shall be in compliance with Annex 2 of JIS X 0221-1:2001. However, OVER LINE (code value 0x7E) specified in JIS X 0201 shall be converted to TILDE (code value 0x007E). Conversion of additional Kanji characters in the region from Row 85, Cell 1 to Row 86, Cell 43 in the additional symbol set (the G set specified by final byte of 03/11) shall be in accordance with Table 5-1. See Annex E.3 of Part 2 for example glyphs of the characters of which code value are to the PUA, and refer to ISO/IEC 10646-1:2000, ISO/IEC 10646-1/Amd.1:2002 and ISO/IEC 10646-2:2001 for all other example glyphs. The Table E-4 of Annex E of Part 2 shall be used for conversion of other additional characters in this set. Character of Row 91, Cell 8 (GL code value 0x7B28, GR code value 0xFBA8) of this area shall be assigned to 0x3012 (POSTAL MARK). For the non-spacing characters of Row 1, Cell 13 to 18 and Row 2, Cell 94 in the Kanji character sets specified in Clause 7.1, Chapter 7, Part 2, Volume 1 of ARIB STD-B24 shall be converted as shown in Table 5-2. The converted characters shall be handled to conform to “ISO/IEC 10646-1:2000 Annex B (normative) List of combining characters.”

Proportional character sets specified in Clause 7.1, Chapter 7, Part 2, Volume 1 of ARIB STD-B24 shall be read as fixed-width character sets. Moreover, mosaic sets, CSI excluding XCS, and C1 control codes shall be ignored.

Table 5-1 Conversion of extension Kanji characters

Row	Cell	UCS code value	Row	Cell	UCS code value	Row	Cell	UCS code value	Row	Cell	UCS code value
85	1	0x3402	85	25	0x5734	85	49	0x693B	85	73	0x73A8
	2	0xE016		26	0xFA10		50	0x6A45		74	0x73C9
	3	0x4EFD		27	0x5880		51	0x6A91		75	0x73D6
	4	0x4EFF		28	0x59E4		52	0xE007		76	0x741B
	5	0x4F9A		29	0x5A23		53	0xE018		77	0x7421
	6	0x4FC9		30	0x5A55		54	0xE019		78	0xE00A
	7	0x509C		31	0x5BEC		55	0xE01A		79	0x7426
	8	0x511E		32	0xFA11		56	0x6BF1		80	0x742A
	9	0x51BC		33	0x37E2		57	0x6CE0		81	0x742C
	10	0x351F		34	0x53AC		58	0x6D2E		82	0x7439
	11	0x5307		35	0x5F34		59	0xE008		83	0x744B
	12	0x5361		36	0x5F45		60	0x6DBF		84	0x3EDA
	13	0x536C		37	0x5FB7		61	0x6DCA		85	0x7575
	14	0x8A79		38	0x6017		62	0x6DF8		86	0x7581
	15	0xE017		39	0xE000		63	0xE009		87	0x7772
	16	0x544D		40	0x6130		64	0x6F5E		88	0x4093
	17	0x5496		41	0x6624		65	0x6FF9		89	0x78C8
	18	0x549C		42	0x66C8		66	0x7064		90	0x78E0
	19	0x54A9		43	0xE006		67	0xE002		91	0xE00B
	20	0x550E		44	0x66FA		68	0xE01B		82	0xE00C
	21	0x554A		45	0x66FB		69	0x7147		93	0xE003
	22	0x5672		46	0x6852		70	0x71C1		94	0xE00D
	23	0x56E4		47	0xE001		71	0x7200			
	24	0x5733		48	0x6911		72	0x739F			

Row	Cell	UCS code value	Row	Cell	UCS code value	Row	Cell	UCS code value	Row	Cell	UCS code value
86	1	0xE004	86	12	0x82AE	86	23	0x8AF6	86	34	0x9348
	2	0x79DA		13	0xE00E		24	0x8DCE		35	0x9592
	3	0x7A1E		14	0x84DC		25	0xE012		36	0x96DE
	4	0x7B7F		15	0xE00F		26	0x8FF6		37	0xE014
	5	0x7C31		16	0x8559		27	0x90DD		38	0x9940
	6	0x4264		17	0x85CE		28	0x9127		39	0x9AD9
	7	0x7D8B		18	0xE010		29	0xE013		40	0xE015
	8	0x7FA1		19	0x87EC		30	0x91B2		41	0x9DD7
	9	0x8118		20	0x880B		31	0x9233		42	0x9EB4
	10	0x813A		21	0x88F5		32	0x9288		43	0x9EB5
	11	0xE005		22	0xE011		33	0x9321			

Table 5-2 Conversion Table for non-spacing characters

Row/Cell	Same grlyph character	UCS code value	UCS Character name
1-13	ACUTE ACCENT	0x0341	COMBINING ACCUTE TONE MARK(Vietnamese)
1-14	GRAVE ACCENT	0x0340	COMBINING GRAVE TONE MARK(Vietnamese)
1-15	DIAERESIS	0x0308	COMBINING DIAERESIS(Dialytika)
1-16	CIRCUMFLEX ACCENT	0x0302	COMBINING CIRCUMFLEX ACCENT
1-17	OVERLINE	0x0305	COMBINING OVERLINE
1-18	LOW LINE	0x0332	COMBINING LOW LINE
2-94	LARGE CIRCLE	0x20DD	COMBINING ENCLOSING CIRCLE

For Kanji characters listed in Table 5-3, the characters for substitution given by XCS shall be converted.

Table 5-3 Characters that call out the substitution character line

GL code value	Plane-Row-Cell	GL code value	Plane-Row-Cell	GL code value	Plane-Row-Cell	GL code value	Plane-Row-Cell
3-2E22	1-14-2	3-7825	1-88-5	4-2121	2-1-1	4-2544	2-5-36
3-2E38	1-14-24	3-787E	1-88-94	4-212B	2-1-11	4-2547	2-5-39
3-2E49	1-14-41	3-7927	1-89-7	4-212E	2-1-14	4-2555	2-5-53
3-2E50	1-14-48	3-7929	1-89-9	4-2136	2-1-22	4-2556	2-5-54
3-2E63	1-14-67	3-7933	1-89-19	4-2146	2-1-38	4-257E	2-5-94
3-2E68	1-14-72	3-7934	1-89-20	4-2170	2-1-80	4-2830	2-8-16
3-2E6E	1-14-78	3-7937	1-89-23	4-2177	2-1-87	4-2837	2-8-23
3-2F2C	1-15-12	3-7938	1-89-24	4-2179	2-1-89	4-2838	2-8-24
3-2F2F	1-15-15	3-7939	1-89-25	4-2322	2-3-2	4-283A	2-8-26
3-2F36	1-15-22	3-793B	1-89-27	4-2325	2-3-5	4-283B	2-8-27
3-2F42	1-15-34	3-793F	1-89-31	4-2327	2-3-7	4-283F	2-8-31
3-2F4C	1-15-44	3-7940	1-89-32	4-2331	2-3-17	4-2840	2-8-32
3-2F5A	1-15-58	3-7947	1-89-39	4-2332	2-3-18	4-2845	2-8-37
3-2F5E	1-15-62	3-794D	1-89-45	4-2338	2-3-24	4-2848	2-8-40
3-2F60	1-15-64	3-7951	1-89-49	4-233F	2-3-31	4-284A	2-8-42
3-2F7B	1-15-91	3-7954	1-89-52	4-2341	2-3-33	4-284B	2-8-43
3-4F61	1-47-65	3-7964	1-89-68	4-234A	2-3-42	4-285B	2-8-59
3-4F62	1-47-66	3-796E	1-89-78	4-2352	2-3-50	4-2866	2-8-70
3-4F63	1-47-67	3-7A2E	1-90-14	4-2353	2-3-51	4-286C	2-8-76
3-4F6E	1-47-78	3-7A33	1-90-19	4-2359	2-3-57	4-2C22	2-12-2
3-7450	1-84-48	3-7A3A	1-90-26	4-235C	2-3-60	4-2C2B	2-12-11
3-745C	1-84-60	3-7A44	1-90-36	4-2377	2-3-87	4-2C30	2-12-16
3-745E	1-84-62	3-7A5D	1-90-61	4-242A	2-4-10	4-2C50	2-12-48
3-7461	1-84-65	3-7B27	1-91-7	4-2431	2-4-17	4-2C65	2-12-69
3-7528	1-85-8	3-7B33	1-91-19	4-2432	2-4-18	4-2C6D	2-12-77
3-752B	1-85-11	3-7B49	1-91-41	4-243A	2-4-26	4-2C72	2-12-82
3-753A	1-85-26	3-7B6C	1-91-76	4-243D	2-4-29	4-2D24	2-13-4
3-7543	1-85-35	3-7B6F	1-91-79	4-2459	2-4-57	4-2D29	2-13-9
3-7565	1-85-69	3-7B79	1-91-89	4-245C	2-4-60	4-2D2A	2-13-10
3-7572	1-85-82	3-7C2F	1-92-15	4-245E	2-4-62	4-2D32	2-13-18
3-7624	1-86-4	3-7C30	1-92-16	4-2463	2-4-67	4-2D34	2-13-20
3-7629	1-86-9	3-7C38	1-92-24	4-246A	2-4-74	4-2D35	2-13-21
3-7632	1-86-18	3-7C49	1-92-41	4-246B	2-4-75	4-2D39	2-13-25
3-7660	1-86-64	3-7C51	1-92-49	4-2472	2-4-82	4-2D56	2-13-54
3-7669	1-86-73	3-7C59	1-92-57	4-2474	2-4-84	4-2D7D	2-13-93
3-7677	1-86-87	3-7D63	1-93-67	4-2475	2-4-85	4-2E23	2-14-3
3-7725	1-87-5	3-7D76	1-93-86	4-2525	2-5-5	4-2E24	2-14-4

3-7755	1-87-53	3-7D7B	1-93-91	4-2532	2-5-18	4-2E3A	2-14-26
3-776C	1-87-76	3-7E66	1-94-70	4-253E	2-5-30	4-2E3C	2-14-28

GL code value	Plane-Row-Cell	GL code value	Plane-Row-Cell	GL code value	Plane-Row-Cell	GL code value	Plane-Row-Cell
4-2E3D	2-14-29	4-7039	2-80-25	4-7427	2-84-7	4-776B	2-87-75
4-2E42	2-14-34	4-7053	2-80-51	4-742E	2-84-14	4-776E	2-87-78
4-2E43	2-14-35	4-707B	2-80-91	4-742F	2-84-15	4-7773	2-87-83
4-2E44	2-14-36	4-712E	2-81-14	4-7434	2-84-20	4-7829	2-88-9
4-2E47	2-14-39	4-7130	2-81-16	4-7435	2-84-21	4-782A	2-88-10
4-2E49	2-14-41	4-7135	2-81-21	4-743D	2-84-29	4-782C	2-88-12
4-2E55	2-14-53	4-7144	2-81-36	4-7442	2-84-34	4-7834	2-88-20
4-2E56	2-14-54	4-715D	2-81-61	4-744F	2-84-47	4-783C	2-88-28
4-2E57	2-14-55	4-7161	2-81-65	4-7450	2-84-48	4-783E	2-88-30
4-2E5B	2-14-59	4-7166	2-81-70	4-7469	2-84-37	4-7842	2-88-34
4-2E77	2-14-87	4-7169	2-81-73	4-746B	2-84-75	4-7856	2-88-54
4-2E78	2-14-88	4-7175	2-81-85	4-7472	2-84-82	4-7863	2-88-67
4-2F2A	2-15-10	4-7177	2-81-87	4-7475	2-84-85	4-7877	2-88-87
4-2F3F	2-15-31	4-717A	2-81-90	4-7479	2-84-89	4-7879	2-88-89
4-2F40	2-15-32	4-7221	2-82-1	4-7535	2-85-21	4-787A	2-88-90
4-2F42	2-15-34	4-7223	2-82-3	4-753A	2-85-26	4-7925	2-89-5
4-2F43	2-15-35	4-7224	2-83-4	4-7546	2-85-38	4-792F	2-89-15
4-2F4E	2-15-46	4-7228	2-83-8	4-7556	2-84-54	4-7932	2-89-18
4-2F59	2-15-57	4-722C	2-82-12	4-7558	2-85-56	4-7939	2-89-25
4-2F61	2-15-65	4-723D	2-82-29	4-755A	2-85-58	4-7942	2-89-34
4-2F69	2-15-73	4-7248	2-82-40	4-755D	2-85-61	4-7948	2-89-40
4-2F6A	2-15-74	4-725B	2-82-59	4-755F	2-85-63	4-7959	2-89-57
4-2F70	2-15-80	4-7275	2-82-85	4-7563	2-85-67	4-795E	2-89-62
4-2F75	2-15-85	4-7276	2-82-86	4-756A	2-85-74	4-7966	2-89-70
4-6E23	2-78-3	4-7332	2-83-18	4-7570	2-85-80	4-7969	2-89-73
4-6E34	2-78-20	4-733D	2-83-29	4-7573	2-85-83	4-796B	2-89-75
4-6E49	2-78-41	4-733E	2-83-30	4-7574	2-85-84	4-797A	2-89-90
4-6E5C	2-78-60	4-7340	2-83-32	4-7575	2-85-85	4-797E	2-89-94
4-6E5E	2-78-62	4-7352	2-83-50	4-7644	2-86-36	4-7A21	2-90-1
4-6E5F	2-78-63	4-735D	2-83-61	4-764E	2-86-46	4-7A2C	2-90-12
4-6E60	2-78-64	4-735E	2-83-62	4-765D	2-86-61	4-7A2F	2-90-15
4-6F32	2-79-18	4-7373	2-83-83	4-7675	2-86-85	4-7A4F	2-90-47
4-6F47	2-79-39	4-7374	2-83-84	4-767E	2-86-94	4-7A50	2-90-48
4-6F4D	2-79-45	4-7375	2-83-85	4-7721	2-87-1	4-7A57	2-90-55
4-6F61	2-79-65	4-7377	2-83-87	4-7722	2-87-2	4-7A65	2-90-69
4-6F64	2-79-68	4-737B	2-83-91	4-7733	2-87-19	4-7A66	2-90-70
4-7022	2-80-2	4-737D	2-83-93	4-7736	2-87-22	4-7A71	2-90-81
4-7029	2-80-9	4-7422	2-84-2	4-7764	2-87-68	4-7A72	2-90-82
4-7033	2-80-19	4-7424	2-84-4	4-7765	2-87-69	4-7A7E	2-90-94

GL code value	Plane-Row-Cell	GL code value	Plane-Row-Cell	GL code value	Plane-Row-Cell	GL code value	Plane-Row-Cell
4-7B21	2-91-1	4-7B5D	2-91-61	4-7C59	2-92-57	4-7E25	2-94-5
4-7B2C	2-91-12	4-7B61	2-91-65	4-7C5D	2-92-61	4-7E29	2-94-9
4-7B2D	2-91-13	4-7B65	2-91-69	4-7C76	2-92-86	4-7E2B	2-94-11
4-7B36	2-91-22	4-7B67	2-91-71	4-7D2C	2-93-12	4-7E32	2-94-18
4-7B37	2-91-23	4-7B69	2-91-73	4-7D4B	2-93-43	4-7E35	2-94-21
4-7B3D	2-91-29	4-7B71	2-91-81	4-7D4C	2-93-44	4-7E53	2-94-51
4-7B3E	2-91-30	4-7C22	2-92-2	4-7D59	2-93-57	4-7E58	2-94-56
4-7B4E	2-91-46	4-7C23	2-92-3	4-7D5B	2-93-59	4-7E5A	2-94-58
4-7B4F	2-91-47	4-7C38	2-92-24	4-7D5D	2-93-61	4-7E6E	2-94-78
4-7B57	2-91-55	4-7C42	2-92-34	4-7D67	2-93-71	4-7E70	2-94-80
4-7B5A	2-91-58	4-7C4C	2-92-44	4-7D6D	2-93-77	4-7E72	2-94-82
4-7B5C	2-91-60	4-7C56	2-92-54	4-7D70	2-93-80	4-7E76	2-94-86

Prefix of “3” and “4” of GL code values represent the level of JIS that specifies that character, i.e., “3” stands for Level 3 and “4” for Level 4.

5.3 Shift JIS

Shift JIS character code defined in Clause 7.3 of Chapter 7, Part 2, Volume 1 of ARIB STD-B24 shall be used.

5.4 EUC-JP

EUC-JP character code defined in Clause 4.1 of Chapter 4, Volume 2 of ARIB STD-B24 shall be used.

Chapter 6 Description command coding

6.1 Geometric

Geometric description command graphic coding shall be in compliance with ARIB STD-B5 and by the method described in Clause 8.1, Chapter 8, Part 2, Volume 1 of ARIB STD-B24.

Chapter 7 Subtitles and character superimpose coding

7.1 Subtitles and character superimpose

Subtitles and character superimpose coding shall be by the method described in Part 3, Volume 1 of ARIB STD-B24.

<Blank Page>

Part 2

Application Execution Engine Platforms

<Blank Page>

Part 2. Application Execution Engine Platforms Contents

Chapter 1 Introduction

1.1 Purpose

This standard stipulates the Application Execution Engine Platforms with respect to the data broadcasting carried out by the digital broadcasting system regulated as the Japanese standard system.

1.2 Scope

This standard applies to data broadcasting in the Application Execution Engine Platforms among the data broadcasting services carried out by digital broadcasting.

1.3 References

1.3.1 Reference Document

This standard refers to the following standard under the abbreviated name MHP1.0

> ETSI TS 101 812, V 1.3.1 Digital Video Broadcasting (DVB);
> Multimedia Home Platform version 1.0.3

This standard also refers to the following standard under the abbreviated name GEM1.0.

> ETSI TS 102 819 V1.2.1 (2003-01) Digital Video Broadcasting (DVB);
> Globally Executable MHP (GEM) Specification 1.0.1

Refer to standards in “2 References” of GEM 1.0 when necessary.

Note):
This standard was prepared based on the international standards GEM1.0 and MHP1.0. However, this standard contains descriptions prior to the revisions of GEM1.0 and MHP1.0, which are now being prepared, in order to maintain consistency with these revisions. In future, when GEM and MHP are formally revised, this standard will be assumed to have been revised accordingly. When any standard referred to in GEM and MHP is revised, such a revision is subject to each standard that is referred to and will not be within the scope of this standard.

1.4 Definitions and abbreviations

1.4.1 Definitions

ARIB-AE: The abbreviated name of this standard is ARIB-AE.

ARIB-J: The abbreviated name of the Application Execution Engine Platforms defined in this standard is ARIB-J. ARIB-J is defined by extending GEM1.0.

BML(Broadcast Markup Language): The applied language defined by the standard ARIB

STD-B24 is defined as BML.

Application: The unit of contents that comprises the data broadcasting service to be carried out by the virtual machine on the receiver.

GEM Application: An application defined as “GEM application” in GEM 1.0. GEM Application operates properly in ARIB-J.

ARIB-J Application: An application based on this standard which is enhancenment of GEM.

ARIB-SI: Service Information stipulated in ARIB STD-B10 to differentiate from SI stipulated in DVB

1.4.2 Abbreviations and terms

AIT: (Application Information Table) This includes the data that transmit additional application information necessary to specify and control applications.

DRCS: (Dynamically Redefinable Character Sets) A system that sends dynamically redefinable non-standard characters by pattern.

DSM-CC: (Digital Storage Media Command and Control) A control system to access files and streams in the digital interactive service defined in ISO/IEC 13818-6.

EIT: (Event Information Table) This includes data relevant to events and programs such as names of events (program), start time and/or duration .

EPG: (Electronic Program Guide) Electronic program schedule.

ES: (Elementary Stream) A basic stream that includes videos, sounds and private data. One ES is transmitted by a PES packet that has an identical Stream ID.

JMF: (Java Media Framework) API for media control in Java.

NPT: (Normal Play Time) Absolute coordinate on the time base that indicates the position of an event generated on the stream.

PSI: (Program Spesific Information) Information source that provides information to realize the fuction for the receiver to cancel the multiplexing and decode various program streams that have been multiplexed by the transmission control information standardized by ISO/IEC 13818-1.

SI: (Service Information) Program Service Information defined in ARIB STD-B10.

UCS: (Universal (Coded) Character Set) International Character Code formulated by ISO (ISO/IEC 10646).

URI: (Uniform Resource Identifier) Method of addressing to objects on the Internet.

UTF: (UCS Transformation Format) Transformation systems of UCS.

Event: A collection of broadcasting data stream elements that belong to the same service with a fixed starting time / ending time. An event is one program such as news, drama or the like.

Event Handler: User-defined function activated by the event that is generated by key input or signal transmission.

Virtual Machine: A computation device defined independently of specific hardware or software.

Service: A series of programs edited by the broadcasting company that can be broadcast as part of a schedule.

Section: A syntax form used for mapping of service information etc. onto a transport stream packed under ISO/IEC 13818-1.

Data Carousel: A system defined in ISO/IEC 13818-16. This distributes data repeatedly in order to download various data by broadcasting.

Module: Unit of data divided by block to be transmitted using a data carousel.

Resource: A network object or service uniquely identified on the network.

Local Event: Part of an event that is broken up by time base or program component.

Please refer to Chapter 3 as well for the terms.

Chapter 2 Format

Each chapter of this standard Part 2 has an assigned number according to the chapter constitution of GEM1.0 to which this standard shall conform. Accordingly, sections that are void in the conforming standard and unnecessary in this standard Part 2. are also specified as void.

Chapter 3 Terms remarks

3.1 javadoc

The detailed definition sections of ARIB-J API in the Annexs are automatically generated using a tool called javadoc from the source code described in Java language. In addition, the terms used in the sections are generally used in Java. As a result, the usage of the terms in the detailed definition sections of ARIB-J API may differ from the terms defined in 1.4.2. or other ARIB standards. The terms to be noted are shown below.

Event: Generic term for java.util.EventObject class and its sub class, and their instances or events that are the factors of these objects' generation. Note that this definition is different from the "Event" defined in 1.4.2. In 1.4.2, the Event is depicted as a program (except for the case of unique terms such as "Expanded format event descriptor".

Super Interface: Synonymous with host interface.

Parameter: Synonymous with argument.

Chapter 4 Standard referencing rule

This chapter describes the arrangement concerning the reference standards. The reference standards referred to in this standards are provided in the Annex C according to the following conditions. This standard conforms to GEM1.0.

Some reference standards are uniquely identifiable by publication date, edition number, version number or so and others are not identifiable.

As for the references that are not uniquely identifiable, new versions of such standards are not applicable.

If the original of a referenced standard is not identifiable, the latest version referenced is applicable.

Amendments or corrigenda that are attached to ISO standards and the like are applicable.

ISBN numbers are considered to be uniquely identifiable standards.

References to RFC are considered to be uniquely identifiable standards. Disuse of the RFC by other RFC is not applicable.

Access to electronic documents by URL reference is considered as a matter of convenience. The existence of the standard to be referenced is not guaranteed.

Chapter 5 Basic architecture

As provided in GEM 1.0, the basic architecture provided in MHP1.0 is an assumed architecture of the terminal to be mounted complying with this standard. This is handled as reference information. Specifications of the preferable receiver for the assumed operation are separately provided. The chapters are not bound by the basic architecture of MHP1.0

Chapter 6 Transport protocol

6.1 Preface

This section conforms to GEM1.0 “6.1 Introduction”.

6.2 Broadcasting Channel Protocol

This chapter provides only the broadcasting channel protocol compliant with GEM1.0. Other protocols and relevant API are extended in the Annex H.

6.3 MPEG-2 Transport Stream

This section conforms to GEM 1.0 “6.2.1 MPEG-2 transport stream”.

6.4 MPEG-2 Section

This section conforms to GEM 1.0 “6.2.2 MPEG-2 sections”.

6.5 DSM-CC Private Data

This section conforms to GEM1.0 “6.2.3 DSM-CC private data”.

6.6 DSM-CC Data Carousel

This section conforms to GEM1.0 “6.2.4 DSM-CC data carousel”.

6.7 DSM-CC U-U Object Carousel

For broadcast transport protocol that transmits the ARIB-J application contents, two systems, i.e. the object carousel and data carousel are specified. Each system conforms to GEM1.0 “6.2.5 Object Carousel” and is accessible via org.dvb.dsmcc package. For details of the transmission protocol, see “Carousel” in the Table 15-1 of Chapter 15.

In addition, for mapping of the data carousel and object carousel in the transmission protocol, see the Annex B. For mapping on org.dvb.dsmcc API in case of using the data carousel, see the Annex P.

Note)
As for the broadcast transport protocol, there are two systems, i.e., the object carousel specified in ISO/IEC 13818-6 and the data carousel specified in Part 3 ARIB ATD-B24. When either of the systems is employed, it should be compliant with GEM1.0 “Annex P: Broadcast transport protocol access” and org.dvb.dsmcc package is used for the broadcasting transport protocol access. With this, transport protocol differences in the low-order layers are absorbed and file accesses are realized in the transmission protocols by common API. In order to realize the above process, expansion of the data carousel is provided in the Annex B of this standard.

6.8 IP multicast transport protocol over broadcasting channel

This section conforms to GEM1.0 “6.2.6 Protocol for delivery of IP multicast over the broadcast channel”.

Note that IP multicast transmission in the broadcasting channel is not supported in this standard and the corresponding functions are not specified. In this connection, the corresponding functions relevant to GEM 1.0 “6.2.10 IP signaling” are not specified.

6.9 IP protocol

This section conforms to GEM1.0 “6.2.7 Internet Protocol(IP)”.

6.10 UDP protocol

This section conforms to GEM 1.0 “6.2.8 User Datagram Protocol (UDP)”.

6.11 Service information(SI)

This shall conform to ARIB STD-B10. This standard does not specify the service information.

However, as for AIT (Application Information Table), which transmits application signals and other Annexs required for ARIB-J application transmission, see Chapter 10 “Application signalling”

6.12 Interactive Communication Protocol

This section conforms to GEM1.0 “6.3 Interaction channel protocols”.

Chapter 7 Content formats

See Part 1 of this standard “Monomedia Coding System in the Application Execution Engine Platforms”.

Chapter 8 Void

Chapter 9 Application model

This chapter specifies the execution control of the ARIB-J application, such as boot and end.

9.1 ARIB-J application

This section conforms to GEM1.0 “9.1 Broadcast GEM Applications”. The ARIB-J application is defined as an extended application of the GEM application by adding the definitions of this standard. And the GEM application is a subset of the ARIB-J application. Therefore, the ARIB-J application created without the additional definitions by this standard is equivalent to the GEM application.

9.2 ARIB-J model

This section conforms to the details of GEM1.0 “9.2 DVB-J model”. In this standard, this model is used as the ARIB-J model .

9.3 How to handle DVB-HTML model

This shall conform to the details of GEM1.0 “9.3 DVB-HTML model”.

Note)
In MHP1.0, DVB-HTML is not specified in detail but only the model is specified. Therefore, the chapter provides only a reference model.

9.4 Resource management between applications

This section conforms to the details of GEM1.0 “9.4 Inter-application resource management”.

Chapter 10 Application signalling

10.1 General

To realize the execution of the ARIB-J application, it is necessary to specify the application and transmit the additional application information to control the application. In GEM1.0, however, the information is provided in the form of an abstract. Therefore, the specific transmission protocol is supposed to be provided on the basis of each standardizing group.

This chapter describes the transmission protocol of the application information to be used in this standard according to GEM1.0 “10 Application signaling”.

In this standard, the additional definitions to ARIB STD-B10 are as follows:

Additional definitions to ARIB STD-B10

- Definition of identification values corresponding to ARIB-J and AIT to identify the data coding scheme
- To store of additional_arib_j_info() into additional identifying information in the data component descriptor for ES that transmits ARIB-J in PMT.
- To store arib_j_info() into additional information inside the data contents descriptor to be stored within the descriptor area of the program event that uses the ARIB-J Application in EIT.
- To store of ait_identifier_info() into additional information inside the data component descriptor for ES that transmits AIT of PMT.
- Assignment of an object carousel to ‘10’ with the field of additional_arib_j_info() and arib_j_info().

With the additional definitions above, the Application Information Table specified in 10.15.1 will be transmitted as an ES that comprises the program in private section mode. Using this AIT, application information is stored. See “10.15.4 Descriptors for AIT (application information table)” for the structure of the descriptor groups to be stored in AIT.

10.2 Common application signal

This section shall conform to the requirements specified in GEM1.0 “11.1.1 Summary of Requirements on Common Signaling”. Specific transmission protocols are described below:
Application information is transmitted in private section mode assuming the application information table (AIT) specified in 10.15.1 as ES that comprises the program. In this way, additional information required for each application is transmitted.

In addition, data coding scheme identification values are assigned in order to indicate the

existence of AIT transmission as well as the existence of ARIB-J application. And the structure within the selector area of the data component descriptor are additionally specified respectively. See 10.6 for details.

The descriptors shown below shall be stored in the AIT at least as common information regardless of the application form.

- Transport protocol descriptor
  - All applications shall be in the scope of at least one transport protocol descriptor. This descriptor can be stored both in a common information descriptor loop and an application information descriptor loop.
- Application descriptor
  - A single application descriptor shall be stored in an application information descriptor loop for each application.
- Application name descriptor
  - A single application shall be stored in an application information descriptor loop for each application.

10.3 Additional application signal necessary for ARIB-J

The requirements specified in GEM1.0 “10.1.2 Summary of additional signaling for DVB-J applications” shall be observed. ARIB-J and DVB-J under this standard are equivalent. Necessary information such as the ARIB-J application shall be transmitted vie AIT. See 10.8 for details.

In the AIT application information descriptor ARIB-J loop, the two descriptors shown below shall be stored at least for an application.

- ARIB-J application descriptor
- ARIB-J application location descriptor

10.4 Additional definitions in PSI/SI

As for application information, application information table (AIT) specified in 10.15.1 is transmitted in private section mode as an ES that comprises the program.

Additional definitions of application signalling for ARIB STD-B10 are shown as follows:

Definition of the identification values corresponding to ARIB-J and AIT to identify the data coding scheme

To store of additional_arib_j_info() into additional identifying information in the data component descriptor for ES that transmits ARIB-J of PMT.

To store of arib_j_info() into additional information inside the data contents descriptor to be stored within the descriptor area of a program event that uses the ARIB-J Application in

EIT.

To store ait_identifier_info() into additional information inside the data component descriptor for ES that transmits AIT of PMT.

Differentiation from other transmission protocols by assigning 0x0004[Value : T.B.D.] as the protocol_id that corresponds to the data carousel. As for the details of the selector_byte, see “10.15.4.2 Transport protocol descriptor”.

Assignment of an object carousel to ‘10’ in the field of additional_arib_j_info()and arib_j_info() to identify the contents transmission protocol on the PAT/PMT level.

In case of transmission_format=’10’(= Object carousel), Association_tag descriptor (tag value: 0x14), deffered_Association_tag descriptor(tab value: 0x15), or Carousel_id descriptor (value: 0x13) specified in ISO/IEC 13818-6 are required to be stored in PMT as needed.

10.5 Data coding scheme identification

A data coding scheme ID (data_component_id) is assigned to the ARIB-J application. Meanwhile, a value of the data coding scheme ID is assigned to AIT transmission and an elementary stream to be transmitted in private section mode is added to PMT.

10.6 Data component descriptor and data contents descriptor

From the data carousel that transmits the ARIB-J application, indirect referencing with a data component descriptor relevant to the ARIB-J coding system and the data contents descriptor shall be made by the component tag (component_tag.)

(1) Data component descriptor in ARIB-J application

When the data coding identification is made by the ARIB-J coding system, the additional_arib_j_info() structure as shown in Table 10-1 is described within the area of additional identification information in the data component descriptor. Additional information that is not transmitted in AIT is stored here.

See ARIB STD-B10 Part 2 “6.2.20 Data component descriptor” for the structure of the data component descriptor.

Table 10-1 additional_arib_j_info()

Data structure	Bit rate	Bit string
additional_arib_j_info(){		
transmission_format	2	bslbf
application_identifier_flag	1	bslbf
document_resolution	4	bslbf
independent_flag	1	bslbf
if (application_identifier_flag == 1) {		
application_identifier()		bslbf
}		
if (transmission_format == '00') {		
download_id	32	uimsbf
ondemand_retrieval_flag	1	bslbf
file_storable_flag	1	bslbf
event_section_flag	1	bslbf
reserved_future_use	5	bslbf
} else if (transmission_format == '01') {		
reserved_future_use	8	bslbf
} else if (transmission_format == '10') {		
carousel_id	32	uimsbf
ondemand_retrieval_flag	1	bslbf
file_storable_flag	1	bslbf
event_section_flag	1	bslbf
reserved_future_use	5	bslbf
}		
}		

Description of additional_arib_j_info( ):

transmission_format (transmission format): The two-bit area specifies the transmission protocol of the ARIB-J application.

Value	Description
00	Data carousel and event message (Except for data service for storage only)
01	Data carousel (Data service for storage only)
10	Object carousel
11	Reserved for future extension.

application_identifier_flag (application identifier flag): This one-bit flag indicates whether the application identifier is included within the selector area.

Value	Description
0	Application identifier is included.
1	Application identifier is not included.

document_resolution (document resolution): ARIB-J application resolution (corresponding to the resolution characteristics) and aspect ratio (corresponding to the display-aspect-ratio characteristics) are indicated. Choose one of the following.

Value	Description
0000	ARIB-J applications with multiple sizes and resolutions are included.
0001	1920 x 1080 (16: 9)
0010	1280 x 720 (16: 9)
0011	960 x 540 (16: 9)
0100	720 x 480 (16: 9)
0101	720 x 480 (4: 3)
0110-1111	Reserved for future extension.

independent_flag(independent listening and viewing availability flag): This indicates whether the data-broadcasting program is assumed to be listened to and viewed independently.

Value	Description
0	Independent listening and viewing impossible
1	Independent listening and viewing possible

application_identifier()(application identifier): A value to uniquely identify the application. It has the following structure. See “10.15.2.1 Coding of Application Identifier” for the details.

Data structure	Bit rate	Bit strings	Description
application_identifier() {			
organization_id	32	bslbf	
application_id	16	bslbf	
}			

organization_id (organization ID): This 32-bit field shows the system that created the application. The ID stores the internationally unique number that has been assigned.

application_id (application ID): This 16-bit field stores the number that is uniquely assigned in the system to identify the application.

If the application described by the descriptor is an additional service to a TV program or a radio program, it is used to specify the application that actually associates with the TV program or the radio Program.

download_id (download ID): This 32-bit field serves as a label that uniquely identifies the carousel. This shows the carousel that should be mounted by default.

ondemand_retrieval_flag (on demand reception for listening and viewing availability flag): This one-bit area indicates, for the application reception transmitted by the said ES, whether the application acquisition from the carousel in each case of the audience operation is assumed. The receivability is regulated by the operation of each media entity.

Value	Description
0	On demand reception for listening and viewing is not available.
1	On demand reception for listening and viewing is available.

file_storable_flag(file storable flag): This indicates whether file storage of the corresponding data broadcasting program is possible. For example, file storage is difficult if the information is updated during the program. The storability is regulated by the operation of each media entity.

Value	Description
0	File not storable
1	File storable

event_section_flag (event section transmission section flag): This one-bit field indicates whether the event message is distributed by this component.

0: The event message is not distributed.

1: The event message is distributed.

carousel_id (carousel ID): This 32-bit field is the identification value that uniquely specifies the object carousel. This identification value is specified by the carousel_id descriptor (carousel identifier descriptor) that is stored in PMT.

(2) Data contents descriptor in ARIB-J application

If the data coding identification is made by the ARIB coding system, the arib_j_info() structure shown in Table 10-2 shall be described in the selector area of the data contents descriptor in EIT. This enables advanced notification of the ARIB-J application to be scheduled for use by the program event unit.

Information concerning ARIB-J application and control signals is stored in AIT. It is not assumed that the application is controlled by the program event unit. Accordingly, there is no mechnaism in AIT that comprehends the schedule by which the ARIB-J application will be used in advance for each program unit.

See ARIB STD-B10 Part 2 “6.2.28 Data Contents Descriptor” for details of the data contents descriptor.

Table 10-2 arib_j_info()

Data structure	Bit rate	Bit strings
arib_j_info(){		
transmission_format	2	bslbf
reserved_future_use	1	bslbf

Data structure	Bit rate	Bit strings
document_resolution	4	bslbf
default_version_flag	1	bslbf
independent_flag	1	bslbf
application_identifier_flag	1	bslbf
content_id_flag	1	bslbf
associated_application_flag	1	bslbf
reserved_future_use	3	bslbf
update_flag	1	bslbf
ISO_639_language_code	24	bslbf
if (application_identifier_flag == 1) {		
application_identifier()		bslbf
}		
if (content_id_flag==1) {		
content_id	32	uimsbf
content_version	16	uimsbf
}		
if (default_version_flag==0) {		
application_profiles_length	8	uimsbf
for (i=0; I<N; i++) {		
application_profile	16	uimsbf
profile_major_version	8	uimsbf
Profile_minor_version	8	uimsbf
profile _micro_version	8	uimsbf
}		
}		
if (transmission_format == ‘00’) {		
arib_carousel_info()		
ondemand_retrieval_flag	1	bslbf
file_storable_flag	1	bslbf
reserved_future_use	6	bslbf
} else if (transmission_format == ‘01’) {		
arib_stored_carousel_info()		
} else if (transmission_format == ‘10’) {		
arib_object_carousel_info()		
ondemand_retrieval_flag	1	bslbf
file_storable_flag	1	bslbf
reserved_future_use	6	bslbf
}		
}		

Description of arib_j_info( ):

transmission_format (transmission format): This two-bit area specifies ARIB-J application transmission protocol.

Value	Description
00	Data carousel and event message (Excluding data service for storage only)
01	Data carousel (data service for storage only)
10	Object carousel
11	Reserved for future extension.

document_resolution (document resolution): The resolution of the ARIB-J application and display-aspect-ratio are indicated by the following.

Value	Description
0000	ARIB-J applications with multiple sizes and resolutions are included.
0001	1920x1080(16:9)
0010	1280x720(16:9)
0011	960x540(16:9)
0100	720x480(16:9)
0101	720x480(4:3)
0110-1111	Reserved for future extension

default_version_flag (default version use flag): This one-bit flag indicates that the default value specified by the operation is used as a profile for execution of the AEIB-J application that is to be transmitted by the corresponding ES.

Value	Description
0	Do not use the default value for the version number
1	Use the default value for the version number

independent_flag (independent listening and viewing availability flag): This indicates whether the data-broadcasting program is assumed to be listened to and viewed independently.

Value	Description
0	Independent listening and viewing impossible
1	Independent listening and viewing possible

application_identifier_flag (application identifier flag): This one-bit flag indicates whether the application identifier is included within the selector area.

Value	Description
0	Application identifier is not included.
1	Application identifier is included.

content_id_flag (contents ID flag): This one-bit flag indicates whether the contents ID and the contents version are included in the descriptor.

Value	Description
0	The contents ID and the contents version are not included.
1	The contents ID and the contents version are included.

associated_application_flag (associated application flag): This one-bit flag indicates that contents associate with the TV program or radio program when the application described by this descriptor is an additional data service to the TV program or radio program. For an application that is not an additional service, the value should always be 0.

Value	Description
0	This application does not include associating contents or is not an additional service.
1	This application is an additional data service and includes associating contents.

update_flag (update flag): This indicates whether there will be differential distribution to this application in the future.

Value	Description
0	There is no differential distribution to this ARIB-J application.
1	There is differential distribution to this ARIB-J application.

ISO_639_language_code (language code): The language code used for ARIB -J application.

application_identifier( ) (application identifier): Value to uniquely identify the application. It has the following structure. See “10.15.2.1. Application Identifier Coding” for details.

Data structure	Bit rate	Bit strings	Description
application_identifier() {			
organization_id	32	bslbf	
application_id	16	bslbf	
}			

organization_id (organization ID): This 32-bit field indicates the system that prepared the application. This ID stores the internationally unique number that has been assigned.

application_id (application ID): This 16-bit field stores the number that identifies the application. The number is uniquely assigned in the system.

When the application described by this descriptor is an additional service to the TV program or radio program, it is used to specify the application that actually associates with the TV program or radio program.

content_id (contents ID): This 32-bit field is a label to identify the data-broadcasting program and is assigned uniquely in the broadcasting company. In case of data storage, if the content_id has the same value as the one of the previous data-broadcasting program, the data can be overwritten.

content_version (contents version): This 16-bit field indicates the version number among the data-broadcasting program that has an identical contents ID.

application_profiles_length (application profile length specification): It indicates the length of the field that specifies the receiver profile with which the application is executable.

profile_major_version (profile major number): This 8-bit field indicates the major number among the version numbers of receiver profiles that should correspond at least to the relevant ARIB-J application execution.

profile_minor_version (profile minor number): This 8-bit field indicates the minor number among the version numbers of receiver profiles that should correspond at least to the relevant ARIB-J application execution.

profile_micro_version (profile micro number): This 8-bit field indicates the micro number among the version numbers of the receiver profiles that should correspond at least to the relevant ARIB-J application execution.

arib_carousel_info( ): The data structure specified in the Annex C.2 of ARIB STD-B24 Part 3.

ondemand_retrieval_flag (on demand reception for listening and viewing availability flag): This one-bit area indicates, for the application reception transmitted by the said ES, whether the application acquisition from the carousel in each case of the audience operation is assumed. The receivability is regulated by the operation of each media entity.

Value	Description
0	on demand reception for listening and viewing is not available.
1	on demand reception for listening and viewing is available.

file_storable_flag (file storable flag):
This indicates whether file storage of the corresponding data-broadcasting program is possible. For example, file storage will be considered difficult if the information is updated during the program. The storability is regulated by the operation of each media entity.

Value	Description
0	File is not storable
1	File is storable

arib_ stored_carousel_info(): Data structure specified in the Annex C.2 of ARIB STD-B24 Standard Part 3

arib_ object_carousel_info(): Data structure specified in10.6 (4) of this standard.

(3) Data component descriptor for AIT transmission
When the data coding ID is AIT transmission, the ait_identifier_info() structure shown in the Table 10-3 shall be described within the selector area of the data component descriptor in PMT.

See ARIB STD-B10 Part 2 “6.2.20 Data component descriptor” for details of the structure data component descriptor.

Table 10-3 ait_identifier_info()

Data structure	Bit rate	Bit strings
ait_identifier_info(){		
for (i=0; i<N; i++) {		
application_type	16	uimsbf
reserved_future_use	3	bslbf
AIT_version_number	5	uimsbf
}		
}		

application_type (application type): This indicates the value of the application type to be transmitted in AIT. As for DVB, DVB-J is assigned 0x0001. In ARIB-J, the value shall be also 0x0001.

AIT_version_number (AIT version number): The current version_number is stored.

(4) Selector area of data component descriptor for object carousel
When information for the object carousel reception control is inserted within the selector area of the data contents descriptor, the following information shall be added at the position in the selector area specified for each data coding scheme.

Table 10-4 arib_object_carousel_info() structure

Data structure	Bit rate	Bit strings
arib_object_carousel_info(){		
num_of_carousels	8	uimsbf
for(i=0; i< num_of_carousels; i++) {		
association_tag	16	uimsbf
event_section_flag	1	bslbf
reserved_future_use	3	bslbf
Component_size_flag	1	bslbf
default_transaction_id_flag	1	bslbf
default_timeout_DSI_flag	1	bslbf
default_leak_rate_flag	1	bslbf
if (component_size_flag == '1') {		
component_size	32	uimsbf
}		
if (default_transaction_id_flag == '1') {		
transaction_id	32	uimsbf
}		
if (default_timeout_DSI_flag == '1') {		
timeout_value_DSI	32	uimsbf
}		
if (default_leak_rate_flag == '1') {		
leak_rate	22	uimsbf
reserved	2	bslbf
}		
}		
}		

Description:
num_of_carousels (number of carousels): This 8-bit field indicates the number of object carousels included in a round of the loop.

association_tag (association tag): This 16-bit field specifies the component stream, to which the DSI message with the ServiceGatewayInfo of the object carousel stored, by the association tag that is assigned by the association_tag descriptor of PMT.

event_section_flag: With this component, the event message distribution is indicated.

component_size_flag (component size flag): This one-bit field indicates whether the component size is coded in the data structure. When the value of the component_size field is not defined yet, it is not coded.

0: not coded
1: coded

default_transaction_id_flag: This one-bit field indicates whether the transaction ID is coded in the data structure. When the acquisition of DII for optional transaction ID is specified, the transaction ID is not coded.

0: Not coded

1: Coded

default_timeoutDSI_flag: This one-bit field indicates whether the DSI timeout value is coded in the data structure. When the default value defined by the operation is used as the value of DSI timeout, it is not coded.

0: Not coded

1: Coded

default_leak_rate_flag: This one-bit field indicates whether the leak rate is coded in the data structure. When the default value defined by the operation is used as the leak rate value, it is not coded.

0: Not coded

1: Coded

component_size (component size): This 32-bit field indicates the total data size (unit: byte) to be transmitted by the said object carousel.

transaction_id (transaction ID): This is the transaction ID value to be transmitted by the component. Non-coded transaction ID shows the necessity of DSI acquisition that has optional transaction ID.

time_out_value_DSI(DSI timeout value): This 32-bit field indicates the recommended timeout value (unit: millisecond) for the whole DSI section reception of the relevant carousel. When the value is 0xFFFFFFFF, this means there is no recommendable timeout value.

leak_rate (leak rate): This 22-bit field indicates the leak rate of the transport buffer of receiver. The unit is 50 byte/s.

10.7 Locator in application description

This shall conform to GEM1.0 “11.3 Locators Within an Application Description”. The locator description in ARIB shall conform to Chapter 14 System Connections of this standard.

10.8 Application description

This shall conform to GEM1.0 “11. 4 Application description”. The transmission protocol is described in 10.2. The section table structure of the AIT to be transmitted in the private section transmission mode shall be compliant with “10.15.1 Application Information Table (AIT)”. The string field in AIT can be encoded by an 8bit-code as well as UTF-8. For operation, one of them should be applied and mixing both coding must be avoided. On the other hand, the strings acquired by the getName() method such as org.dvb.application API can be automatically converted to strings that are usable on Java.

See “10.9 Transmission and monitoring of application description” for details of additional definitions.

10.9 Transmission and monitoring of application description

This shall conform to GEM1.0 “10.4.1 Application description transmission and monitoring”. As for the process of transmission, the section table that is compliant with the AIT structure is transmitted as ES that comprises the program via private section transmission mode.

For the operation, 0x0001 is assigned as the application_type of ARIB-J and 0x0004 [value: T.B.D] is assigned as the protocol_id value to signify the data carousel.

In addition, the selector_byte in the transport protocol descriptor for the data carousel shall be the same syntax as the case of the “object carousel transport protocol (protocol_id =0x0001)”. See the Table 10-13 for details of the structure.

10.10 Visibility of application description

This shall conform to GEM1.0 “10.4.2 Visibility of application description”.

10.11 Details of application description

Application descriptions are transmitted based on the transmission protocol described in 10.2. The application descriptors specified in “10.15.4. Descriptors for AIT (application information table)” shall be used for storage of application descriptions.

10.12 Application handling from previously selected service

This shall conform to GEM1.0 “10.4.4 Application from previously selected services”.

10.13 ARIB-J specific application description

This shall conform to GEM1.0 “10.5 DVB-J specific application description”.

10.14 Details of ARIB-J application description

ARIB-J application descriptions are transmitted based on the transmission protocol described in 10.2.

The AIT structure as well as the descriptor structure in AIT shall conform to “10.5 Application information coding system”. Note that DVB-J in MHP shall be interpreted as ARIB-J in this standard. The application descriptors specified in “10.15.4.3 Descriptors for ARIB-J application” shall be used for storage of application descriptions of the ARIB-J application.

10.15 Application information coding system

10.15.1 Application information table (AIT)

In AIT (Application Information Table), all information relevant to the application and requirements for start-up status are stored. It is also possible to instruct the receiver to change the start-up status from the TV station side with the data in the AIT.

All AIT sections that have the same Applicaton_Type on the identical PID comprise a sub-table. The descriptor tag value in the AIT shall be unique in the AIT.

The data structure of the application information table is shown in Table 10-5 below.

Table 10-5 Application Information Table

Data structure	Number of bits	Bit string
application_information_section (){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
application_type	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
reserved_future_use	4	bslbf
common_descriptors_length	12	uimsbf
for (i=0, i<N; i++) {		
descriptor ()		
}		
reserved_future_use	4	bslbf
application_loop_length	12	uimsbf
for (i=0; i<N;i++) {		
application_identifier ()		
application_control_code	8	uimsbf
reserved_future_use	4	bslbf
application_descriptors_loop_length	12	uimsbf
for (j=0; j<M; ;j++) {		
descriptor ()		
}		
}		
CRC_32	32	rpchof
}		

Descriptions:

table_id (Table ID): In this 8-bit field, 0x74 is stored to indicate that this is the said table.

section_syntax_indicator (section syntax indicator): The section syntax indication is always “1” in the one-bit field.

section_length (section length): This is a 12-bit field. The first two bits shall always be “00”. This specifies the number of bits from the section length field to the last section including CRC32. The value must be lower than 1021 (0x3FD in hexadecimal).

application_type (application type): This 16-bit field indicates the value of application type that is being transmitted by AIT. In DVB, 0x0001 is assigned to DVB-J application. The value is also 0x0001 in ARIB-J

application_type	Description
0x0000	reserved_future_use
0x0001	ARIB-J Application
0x0002…0x7FFF	Undefined

version_number (version number): This 5-bit field is the version number of the sub-table. Add one to the version number when a change is made in the information inside the sub-table. When the value reaches “31,” the next value will be back to “0”.

current_next_indicator (current next indication): This one-bit indication is always “1”.

section_number (section number): This 8-bit field shows the section number. The section number of the first section in the sub-table is 0x00. Each addition of a section that has the identical table ID and application type adds “1” to the section number.

last_section_number (last section number): This 8-bit field specifies the number of the last section in the sub-table to which the sections belong.

common_descriptors_length (common descriptors loop length): This 12-bit field specifies the byte length of the subsequent common descriptors area. The descriptors inside descriptors area are applicable to all applications in the AIT sub-table.

application_control_code (application control code): This 8-bit field specifies the control code that controls application status. This field is dependent on the value of application type. See “10.15.3 Application life cycle control” for details.

application_loop_length (application information loop length): This 12-bit field specifies the whole loop byte length where subsequent application information is stored.

application_identifier () (application identifier): See table 10-6.

application_descriptors_loop_length (application information descriptors loop length): This 12-bit field specifies the byte length of the subsequent descriptors area. These descriptors in the descriptors area are applicable only to the designated application.

10.15.2 Application ID

10.15.2.1 Coding of application identifier

An application is uniquely identified by the application identifier shown in table 10-6. This identifier is comprised of a 6-byte (48-bit) structure.

Table 10-6 Application Identifier

Data structure	Number of bits	Bit string
application_identifier () {		
organization_id	32	bslbf
application_id	16	bslbf
}		

Description:

organization_id (organization ID): This 32-bit field indicates the organization that created the application. This ID stores the number uniquely assigned in the world.

application_id (application ID): This 16-bit field stores the number that identifies the application and is uniquely assigned in the organization ID.

Application ID is divided into two ranges. One is the unsigned application range and the other is the signed application range. This division is made for security purposes. (See Chapter 12 Security). The range of the value is respectively shown below.

Application ID value	Application type
0x0000…0x3fff	Range for unsigned application
0x4000…0x7fff	Range for signed application
0x8000…0xfffd	(Reserved by DVB)
0xfffe	Wild card value (indicates all signed applications of the same organization ID)
0xffff	Wild card value (indicates all applications of the same organization ID)

In the application ID, the values 0xffff and 0xfffe are for wild card value. These are not used to specify the application, but for example are used as a descriptor for external application authorization. 0xffff corresponds to all applications that have the same organization ID (ornanization_id). 0xfffe corresponds to all signed applications that have the same organization ID.

Sometimes the same application identifier is used between applications of different types, as in the case of the execution of the same function by different application types, for example.

Note that only one type appears in the collection of AIT sub-tables of the same application type in one service.

10.15.2.2 Effect on life cycle

In this section, the basic outline of the effect on life cycle will be presented for the occasion when the service is changed over or the applications that have the same application identifier are started up.

- When the service is changed, if service_bound_flag in the active application in the previous service is “0,” and the application identifier exists in the newly selected service’s AIT, then the application works continuously, unless there is any resource constraint.
- When the service is changed, if service_bound_flag in the active application in the previous service is “0,” and if only the application is on the list of external application authorization descriptors, then the application works continuously, even if the application is not a part of the newly selected service, unless there is any resource constraint.
- As for an application that has one application identifier, only one instance is activated. Even if another application has the same application identifier, it cannot be activated if one application with the same application identifier has already existed. This affects the behavior of application start API or automatic start application.
- If service_bound_flag “1” is set for the application, the application will end (KILL) at

each service selection.

See “Annex S Enumeration and activation of application” for details.

10.15.2.3 Application ID authentication

This section conforms to GEM1.0 “12.5 Profile of X.509 certificates for authentication of applications”. Application ID is authenticated with organization ID (organization_id) stored in the subject field of certificate.

10.15.3 Application life cycle control

Signaling mechanisms will be provided from the broadcasting station to control the life cycle for the standard type applications.

10.15.3.1 Entering and leaving the application domain

Application domain is defined as a collection of services that has applications listed in AIT. This means the applications are the ones listed in the application information loops of AIT or the ones listed in external application authorization descriptors. The services of which applications are not listed as in the way mentioned above are regarded as outside the application domain.

10.15.3.2 ARIB-J application dynamic control

Dynamic control of the application life cycle is signalized with the application_control_code that is stored in AIT by the application. This control code enables the broadcasting station to inform the receiver how the application life cycle should be handled. The details of the control code are defined for each application type.

If the receiver received a control code that the application cannot identity, the current status will be continued. If a change of control code caused status transition of application, AppStateChangeEvent will be generated for all ARIB-J application that registered the reception of the event.

The definition of the control code for ARIB-J application shall conform to the GEM1.0 “10.4.3 Content of the application description”. The details are shown in table 10-7.

Table 10-7 ARIB-J Application Control Code Value

Code	ID name	Description
0x00		reserved_future_use
0x01	AUTOSTART	File system elements (e.g. module of carousel) that include Xlet interface embedded class are loaded and the Xlet embedded class is read into the virtual machine. Then the instance of Xlet object is created to start the application according to the normal restrictions.
0x02	PRESENT	This indicates how the current application exists in the service. This is, however, not automatically started.
0x03	DESTROY	If the control code is changed from AUTOSTART or PRESENT to DESTORY, the destroy method in Xlet will be called. Note that in this case the destroy method is called in the state where the unconditional parameter is false.
0x04	KILL	If the control code is changed from AUTOSTART or PRESENT to KILL, the destroy method in Xlet is called by the application manager. Note that in this case the destroy method is called in the state where the unconditional parameter is true.
0x05		reserved_future_use
0x06	REMOTE	This indicates remote application. This application can be started only after service selection.
0x07…0xFF		reserved_future_use

See “9.2 ARIB-J model” for details of ARIB-J application execution control.

10.15.4 Descriptors for AIT (application information table).

10.15.4.1 Common descriptors

In this section, descriptors to be used commonly in the AIT regardless of the application type will be described.

(1) Application descriptor

One application descriptor is stored in the AIT application information descriptor loop for each application.

Table 10-8 Application descriptor structure

Data structure	Number of bits	Bit string
application_descriptor () {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
application_profiles_length	8	uimsbf
for (i=0; i<N ; i++) {		
application_profile	16	uimsbf
version.major	8	uimsbf
version.minor	8	uimsbf
version.micro	8	uimsbf
}		
service_bound_flag	1	bslbf
visibility	2	bslbf
reserved_future_use	5	bslbf
application_priority	8	uimsbf
for (i=0; i<N ; i++) {		
transport_protocol_label	8	uimsbf
}		
}		

Description:
descriptor_tag (descriptor tag): In this 8-bit field, 0x00 is stored to indicate that this is the said descriptor.

application_profiles_length (application profile information length): This 8-bit field indicates the whole byte length of application profile information that is included in the subsequent loop.

application_profile (application profile): In this 16-bit field, the application profile that can execute the application is stored. If this profile is mounted in the receiver, this means the application is executable. The details of profile are defined for each application type.

version.major (major version): This 8-bit field indicates the major version of the above mentioned profile.

version.minor (minor version): This 8-bit field indicates the minor version of the above mentioned profile.

version.micro (micro version): This 8-bit field indicates the micro version of the above mentioned profile.

The four field described above comprise the minimum profile for execution of this application. ARIB-AE terminal starts this application if any of the following theoretical formulas is

applicable and any profile that shows “true” exists in the application profile information.

(application profile ∈ collection of profiles mounted in the terminal)
AND {(application major version < major version of the terminal for the profile)
OR [(application major version = major version of the terminal for the profile)
AND ({application minor version < minor version of the terminal for the profile }
OR {[application minor version = minor version of the terminal for the profile]
AND [application micro version ≤ micro version of the terminal for the profile]})] }

Profile coding in ARIB-J application shall conform to GEM1.0 “16.1 System constants”.

service_bound_flag (service bound flag): This one bit field indicates if the application is effective only in the present service. If the field is “1,” the application is relevant only to the present service. When the service is changed to another service, end processing of the said application will be started.

visibility (visibility): This 2-bit field indicates if the application is visible to end users when it is activated. Status definitions to the visibility value are shown below.

Visibility value	Description
00	This application is visible neither to other applications via application enumeration API nor to users via navigator except for error information of log-out and the like.
01	This application is not visible to users but visible from other applications via application enumeration API.
10	reserved_for future_use
11	This application is visible to both users and other applications.

application_priority (application priority): This 8-bit field indicates relative priority between the applications notified in the service.

transport_protocol_label (transport protocol label): This 8-bit field stores the value for unique identification of the transport protocol that transports the application. The value corresponds to transport_protocol_label field value of transport protocol descriptor.

(2) Application name descriptor

One application name descriptor is stored for each application in the AIT application information descriptor loop. The application name is used for differentiation and gives information to users.

Table 10-9 Structure of application name descriptor

Data structure	Number of bits	Bit string
application_name_descriptor () {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for (i=0; i<N ; i++) {		
ISO_639_language_code	24	bslbf
application_name_length	8	uimsbf
for (j=0; j<application_name_length ; j++) {		
application_name_char	8	uimsbf
}		
}		
}		

Description:
descriptor_tag (descriptor tag): In this 8-bit field, 0x01 is stored to indicate this is the said descriptor.

ISO_639_language_code (language code): This 24-bit field identifies the language of the application name descriptor. The language code is indicated by the three-alphabetical -letter code as regulated in ISO 639-2. Each letter is coded in 8-bit according to ISO 8859-1 and inserted according to the order into the 24-bit field.

application_name_length (application name description length): This 8-bit field indicates the byte length of the subsequent application name description.

application_name_char (application name description): This is an 8-bit field and specifies the letter description of the application name. This letter description is used as information for users.

(3) Application icons descriptor

The application icon information descriptor is stored by one at most for each application. The descriptor indicates the information about the icon relevant to the application. The contents format of the icon is coded by PNG and uses the system provided in this standard Part 1 “3.3 PNG.”.

Table 10-10 Structure of application icons descriptor

Data structure	Number of bits	Bit string
application_icons_descriptor () {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
icon_locator_length	8	uimsbf
for (i=0; i<N ; i++) {		
icon_locator_byte	8	uimsbf

Data structure	Number of bits	Bit string
}		
icon_flags	16	bslbf
for (i=0; i<N ; i++) {		
reserved_future_use	8	uimsbf
}		
}		

Description:
descriptor_tag (descriptor tag): This 8-bit field stores 0x0B that indicates the said descriptor.

icon_locator_length (icon locator length): This 8-bit field indicates the byte length of subsequent icon locator.

icon_locator_byte (icon locator): This 8-bit field stores the locator to still picture file as an icon. This locator is to be put before the icon file name. The details shall conform to the GEM1.0 “10.4.3 Content of the application description”. As for ARIB-J application, the details shall conform to the GEM1.0 clause 10.4.3 in the case where application_type is 0x0001 and a relative path from the base directory defined by ARIB-J application is stored.

icon_flags (icon flag): This 16-bit field stores the flag that indicates the size and the aspect ratio of the usable icon. The details conform to the GEM1.0 “10.4.3 Content of the application description” and the coding for each case is as shown below. The value is stored after OR (logical ad) by unit.

icon flag bits	Icon size and aspect ration
0000 0000 0000 0001	32 x 32 for square pixel display
0000 0000 0000 0010	32 x 32 for broadcast pixels on 4: 3 display
0000 0000 0000 0100	24 x 32 for broadcast pixels on 16: 9 display
0000 0000 0000 1000	64 x 64 for square pixel display
0000 0000 0001 0000	64 x 64 for broadcast pixels on 4: 3 display
0000 0000 0010 0000	48 x 64 for broadcast pixels on 16: 9 display
0000 0000 0100 0000	128 x 128 for square pixel display
0000 0000 1000 0000	128 x 128 for broadcast pixels on 4: 3 display
0000 0001 0000 0000	96 x 128 for broadcast pixels on 16: 9 display
xxxx xxx0 0000 0000	reserved_future_use

The file name of icon is coded according to the above described icon flag value. The file name coding method shall conform to the GEM1.0 “10.4.3 Content of the application description”.

(4) External application authorization sescriptor

This descriptor can be stored in one or more in the first common descriptor loop of AIT as needed. In each descriptor, information relevant to external applications are stored. An external application is the ones that can operate continuously with the applications listed in the AIT sub-table but cannot be activated from the said service.

This external authorization is applicable to the application that has application_identifier () in application_type specified by AIT that includes this descriptor.

Table 10-11 Structure of external application authorization descriptor

Data structure	Number of bits	Bit string
external_application_authorisation_descriptor () {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for (i=0; i<N ; i++) {		
application_identifier ()		
application_priority	8	uimsbf
}		
}		

Description:

descriptor_tag (descriptor tag): In this 8-bit field, 0x05 is stored to indicate this is the said descriptor.

application_identifier () (application identifier): This 48-bit field indicates the application with which external reference is available. See “10.15.2 Application ID” for details of the field structure.

application_priority (application priority): This 8-bit field indicates the priority of the present application on the assumption of the said service context. See “10.15.4.1 (1) Application descriptor” for details of the priority.

10.15.4.2 Transport protocol descriptor

(1) Transport protocol descriptor

This descriptor indicates the transport protocol ID relevant to the service component and stores information about the protocol. This descriptor is stored in the first common descriptor loop or application information descriptor loop. When it is stored in the common descriptor loop, it is applicable to the whole sub-tables of AIT. The transport protocol descriptor in the application information descriptor loop describes the additional transport protocol to be used specifically in the application.

The application described in this section must be within the scope of at least one transport protocol descriptor.

Table10-12 Structure of transport protocol descriptor

Data structure	Number of bits	Bit string
transport_protocol_descriptor () {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
protocol_id	16	uimsbf
transport_protocol_label	8	uimsbf
for (i=0; i<N ; i++) {		
selector_byte	8	uimsbf
}		
}		

Description:

descriptor_tag (Descriptor tag): This 8-bit field stores 0x02 that indicates this is the said present descriptor.

protocol_id (protocol ID): This 16-bit field indicates the protocol that transports the application.

Value	Description
0x0000	reserved_future_use
0x0001	Object carousel transport protocol
0x0002	reserved
0x0003	reserved
0x0004 [T.B.D]	Data carousel transport protocol
0x0005…0xFFFF	reserved_future_use

transport_protocol_label (transport protocol label): This 8-bit field indicates the value that uniquely identify the transport protocol in the AIT section. For application descriptor, see the connection device (e.g. elementary stream of carousel) that transports the application with these values.

selector_byte (selector area): This is an 8-bit field and stores additional information provided for each protocol ID. The data structure described in the area is specified separately for each protocol ID.

protocol_id value	Selector area structure
0x0000	reserved_future_use
0x0001	See table 10-13
0x0002	Undefined
0x0003	Undefined
0x0004 [T.B.D]	See table 10-13
0x0005…0xFFFF	reserved_future_use

Data structures are shown below for object carousel transport protocol (protocol_id=0x0001) and data carousel transport protocol (protocol_id=0x0004).

Table 10-13 Structure of transport protocol descriptor delector area
(In case of transport protocol of object carousel / data carousel transport protocol)

Data structure	Number of bits	Bit string
remote_connection	1	bslbf
reserved_future_use	7	bslbf
if (remote_connection == “1”) {		
original_network_id	16	uimsbf
transport_stream_id	16	uimsbf
service_id	16	uimsbf
}		
component_tag	8	uimsbf

Description:
remote_connection (remote connection): If this one-bit field is “1,” this shows that the actual service component is transmitted from another source than the one that transmits to AIT. A service like this is not performed automatically but is visible and possible to start via API as specified in “Annex S Application enumeration and activation of application” In addition, REMOTE is stored to such application in application_control_code.

original_network_id (original network ID): When remote_connection is “1,” the original network ID of the actual service transmission is stored.

transport_stream_id (transport stream ID): When remote_connection is “1,” the transport stream ID of the actual service transmission is stored.

service_id (service ID): When remote_connection is “1,” the service ID of the actual service transmission is stored.

component_tag (component tag): This indicates the main service component that transmits the application. In case of data carousel, the elementary stream that transmits the carousel automatically mounted at the application start is indicated. In case of object carousel, the elementary stream that transmits DSI is indicated.

(2) Pre-fetch descriptor

Only one pre-fetch descriptor is stored in the application information descriptor loop in AIT as needed. This is defined to be used for object carousel (protocol_id=0x0001). Each descriptor is associated to one transport protocol descriptor via transport protocol label.

ARIB-AE terminal may acquire modules denoted by the label in advance so as to speed up the starting time of the application. As for the labels, the label descriptors specified by object carousel are used. See “Carousel” in table 15-1 of Chapter15 for details.
The broadcast of this signaling is optional. Accordingly, the receiver mounting for the pre-fetch process is optional.

Table10-14 Structure of pre-fetch descriptor

Data structure	Number of bits	Bit string
prefetch_descriptor () {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
transport_protocol_label	8	uimsbf
for (i=0; i<N ; i++) {		
label_length	8	uimsbf
for (j=0; j<label_length ; j++) {		
label_char	8	uimsbf
}		
prefetch_priority	8	uimsbf
}		
}		

Description:
descriptor_tag (descriptor tag): In this 8-bit field, 0x0C is stored to show the said descriptor.
transport_protocol_label (transport protocol label): This 8-bit field stores the transport protocol label to specify the object carousel that transmits the modules referred in the above Pre-fetch Descriptor. See " 10.15.4.2 Transport protocol descriptor" for transport protocol label.

label_length (label length): This 8-bit field indicates the byte length of the label description to be included in the consequent loop.

label_char (label description): This is an 8-bit field. A module label is stored. This corresponds to the label description transmitted by the label descriptor that is stored in userInfo of moduleInfo of DII in the object carousel.

prefetch_priority (pre-fetch priority): This 8-bit field stores the values from 1 to 100. These values show the priority of pre-fetch. The higher value shows the higher priority.

(3) DII location descriptor
For each application, only one DII location descriptor can be given as needed. This descriptor can be stored both in the common descriptor loop and application information descriptor loop. This is defined to be used for object carousel (protocol_id=0x0001). Each descriptor is associated with one transport protocol descriptor via transport protocol label.

The module group as a part of DSM-CC object carousel is notified by DownloadInfoIndication (DII). All DII messages that show the existing object carousel cannot be listed all in one at one location.

It is necessary to make all the relevant DII messages available in order to find the modules corresponding to labels for pre-fetch (See “10-15.4.2 (2) Pre-fetch descriptor.). DII location descriptor lists the locations of these DII.

In case the DII location descriptor is not included, only DII indicating modules that include ServiceGateway are found.

The loops of DII identification in the descriptor are located in the order of importance. Namely, DII that has labels with higher pre-fetch priority shall be located at the beginning of the loop. The receiver mounted with module-base pre-fetch mechanism checks DII in the order listed in DII location descriptor.

Table10-15 DII location descriptor structure

Data structure	Number of bits	Bit string
DII_location_descriptor () {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
transport_protocol_label	8	uimsbf
for (i=0; i<N ; i++) {		
reserved_future_use	1	bslbf
DII_identification	15	uimsbf
association_tag	16	uimsbf
}		
}		

Description:
descriptor_tag (descriptor tag): This 8-bit field stores 0x0D that indicates this is the said descriptor.

transport_protocol_label (transport protocol label): This 8-bit field stores the transport protocol label that specifies the object carousel that is transmitting the modules referred in pre-fetch descriptor. See “10.15.4.2 Transport protocol descriptor” for details of transport protocol label.

DII_identification (DII identification): This 15-bit field stores the value that specifies the DIImessage. This value corresponds to the bit 1-15 of the transaction ID in dsmMessageHeader () of DII message.

association_tag (association tag): This 16-bit field indicates the relationship with DII message that is broadcasted (e.g. elementary stream).

10.15.4.3 Descriptors used in ARIB-J

(1) ARIB-J application descriptor

One descriptor is stored in the AIT application information descriptor loop for each ARIB-J application. This indicates the parameter information for application starting.

Table 10-16 Structure of ARIB-J application descriptor

Data structure	Number of bits	Bit string
arib_j_application_descriptor () {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for (i=0; i<N ; i++) {		
parameter_length	8	uimsbf
for (j=0; j<parameter_length ; j++) {		
parameter_byte	8	uimsbf
}		
}		
}		

Description:

descriptor_tag (descriptor tag): In this 8-bit field, 0x03 is stored to indicate the said descriptor.

parameter_length (parameter length): This 8-bit field shows the byte length of the subsequent parameter description.

parameter_byte (parameter description): This is an 8-bit field. The string to be given to the application as parameters is stored.

(2) ARIB-J application location descriptor

The descriptor is stored in AIT application information descriptor loop by one for each ARIB-J application. This stores necessary path information in the application.

Table 10-17 Structure of ARIB-J application location descriptor

Data structure	Number of bits	Bit string
arib_j_application_location_descriptor () {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
base_directory_length	8	uimsbf
for (i=0; i<N ; i++) {		
base_directory_byte	8	uimsbf
}		
classpath_extension_length	8	uimsbf
for (i=0; i<N ; i++) {		
classpath_extension_byte	8	uimsbf
}		

Data structure	Number of bits	Bit string

for (i=0; i<N ; i++) {
initial_class_byte
}
}
	8	uimsbf

Description:
descriptor_tag (descriptor tag): In this 8-bit field, 0x04 is stored to indicate the said descriptor.

base_directory_length (base directory length): This 8-bit field indicated the byte length to be included in the subsequent loop. The stored value shall be 1 or higher.

base_directory_byte (base directory byte): This in an 8-bit field. The directory name from the file system route should be stored by delimiter letter using “/” (0x2F). This directory name is used for the base directory in the relative path. If the route directory of file system is designated as a base directory, “/” should be stored.

classpath_extension_length (additional class path length): This 8-bit field indicates the byte length of subsequent additional class path.

classpath_extension_byte (additional class path): This is an 8-bit field. When a class path is designated by the directory other than base directory, the class path name is stored. The directory name from the file system route is stored by the delimiter letter using “/” (0x2F) . If there are more than one paths, they should be stored by enumerating with “;” (0x3B).

initial_class_byte (initial activation class): This is an 8-bit field. The object name on the file system of Xlet interface mounted class is stored.

10.15.5 Constant value

Table 10-18 Constant value used for application information transmission

Identification name	Form	Value	Transmitted from:	Scope
Data contents descriptor	Descriptor tag (descriptor_tag)	0xC7	EIT	STD-B10
Data coding descriptor		0xFD	PMT	
Carousel ID descriptor		0x13		
Association tag descriptor		0x14		
Extension association tag descriptor		0x15		
Cache priority descriptor	Descriptor tag (descriptor_tag)	0x71	DII module information	STD-B23 (at data carousel transmission
Label descriptor		0x70	DII moduleinfo userInfo	STD-B23 (at object carousel transmission)
Cache priority descriptor		0x71		
Contents style descriptor		0x72	BIOP objectInfo	
(Reserved for OC by MHP)		0x73…0x7F	OC	
Application information table	Table ID (table_id)	0x74		STD-B23
Application descriptor	Descriptor tag (descriptor_tag)	0x00	AIT	STD-B23
Application name descriptor		0x01		
Transport protocol descriptor		0x02		
ARIB-J Application descriptor		0x03		
ARIB-J Application location descriptor		0x04		
External application authorization descriptor		0x05		
(Reserved for future use)		0x06,0x07		
(Reserved by MHP)		0x08…0x0A		
Application icon descriptor		0x0B		
Pre-fetch descriptor		0x0C		
DII location descriptor		0x0D		
(Reserved by MHP)		0x0E…0x7F		
User definition		0x80…0xFE		
ARIB-J coding system	Data coding scheme (data_ component_id)	[T.B.D]	PMT	STD-B23
AIT		[T.B.D]		
Data carousel	Transmission format (transmission_format)	‘01'	Area of data component descriptor	STD-B23
Object carousel		‘10’		
MHP object carousel	Protocol ID (protocol_id)	0x0001	AIT	STD-B23
ARIB data carousel		0x0004 [T.B.D]		
ARIB-J application type	Application type (application_type)	0x0001	AIT	STD-B23

*1 See “B.2.1.3 Caching behavior ” for structure of descriptors.

Chapter 11 ARIB-J platform

This chapter specifies the virtual machine and API that comprise ARIB-J.

11.1 Virtual machine

This section conforms to GEM1.0 “11.1 The virtual machine”.

11.2 General issues

This section conforms to GEM1.0 “11.2 General issues” with following restrictions and enhancements.

With regard to all methods of Event Listener Removal in the jp.or.arib.tv package, nothing will be conducted if the said listener is not registered.

Support for encoding EUC-JP, UCS and SHIFT JIS defined in ARIB STD-B24 shall be added.

11.3 Fundamental ARIB-J-APIs

This section conforms to GEM1.0 “11.3 Fundamental DVB-J APIs”. with following restrictions and enhancements.

The ARIBLocator class shall be adopted as a locator. This class shall be a subclass of org.davic.net.Locator. In addition, ARIBNetworkBoundLocator that inherits ARIBLocator shall be adopted as a class corresponding to DvbNetworkBoundLocator.

11.4 Presentation APIs

This section conforms to GEM1.0 “11.4 Presentation APIs” with following restrictions and enhancements.

11.4.1 Subtitle control API

It is not essential to support the following classes and interfaces under GEM1.0 “11.4 Presentation APIs”. However, this standard includes the following subtitle relating classes and interfaces of the org.dvb.media package.

org.davic.media.SubtitlingLanguageControl
org.dvb.media.SubtitlingEventControl
org.dvb.media.SubtitleAvailableEvent
org.dvb.media.SubtitleListener
org.dvb.media.SubtitleNotAvailableEvent
org.dvb.media.SubtitleNotSelectedEvent
org.dvb.media.SubtitleSelectedEvent

11.4.2 Streamed type media API

In order to correspond to the video stream specifications defined in the ARIB standard, the jp.or.arib.tv.media.ARIBVideoFormatControl interface described in Annex N “Streamed media API extensions” of this standard is additionally defined. In all cases where org.dvb.media.VideoFormatControl is usable under GEM1.0, ARIBVideoFormatControl shall be usable by replacement.

Note 1)

ActiveFormatDescription information is not included in the video stream under the ARIB standard.

Therefore, the org.dvb.media.VideoFormatControl.getActiveFormatDefinition() method always returns AFD_NOT_PRESENT. If you want to acquire the display area information of the contents in which ARIB-J application is included in the video stream, use the ARIBVideoFormatControl.getDisplayVideoSize() method of the jp.or.arib.tv.media package so as to acquire the information.

Note 2)

If you want to acquire video stream format information such as 1080i, 720p, 480i, and 480p from ARIB-J application, use the ARIBVideoFormatControl.getSourceVideoSize() and getProgressiveSequence() methods defined in the jp.or.arib.tv.media package to acquire the information.

Note 3)

Following Figure 11-1 shows the relation between the constant that indicates the video format conversion form defined as DFC_* by org.dvb.media.VideoFormatControl and “Desired display format” specified in ARIB STD-B21 6.1.2.

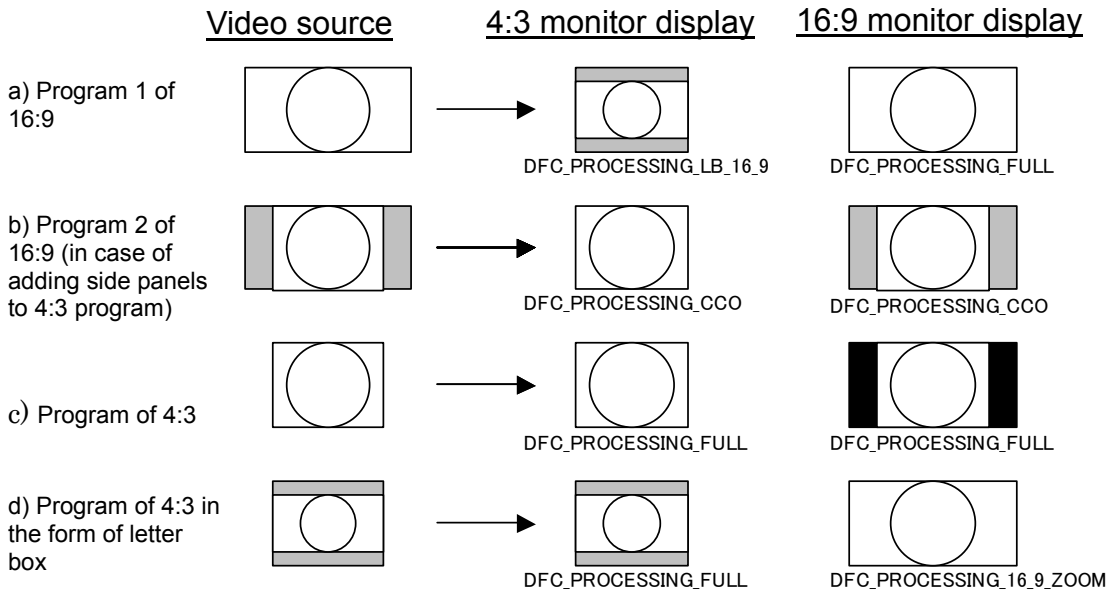


Fig. 11-1 The relation between the VCF constant and desired display format

The following class and interfaces defined in Annex N “Stream-type Media API Extensions” are added as API to detect audio language switching.

jp.or.arib.tv.AudioLanguageEventControl
jp.or.arib.tv.AudioLanguageEventListener
jp.or.arib.tv.AudioLanguageChangedEvent

JMF Player created for a locator to indicate the service must support AudioLanguageEventControl in addition to the JMF Control types desired in GEM1.0.

In addition, the relation between MediaSelectControl defined in GEM1.0 and each LanguageControl (see MHP1.0 “11.4.2.7 Intersection between MediaSelectControl and SubtitlingLanguageControl / AudioLanguageControl”) also applies to AudioLanguageEventControl. In other words, if the audio language is changed by using MediaSelectControl, AudioLanguageChangedEvent will be generated.

11.4.3 CA relevant APIs

This standard includes the following CA-relevant classes of the org.dvb.media package, while GEM1.0 “11.4 Presentation APIs” does not require support of the classes.

org.dvb.media.CAStopEvent
org.dvb.media.CAException

11.5 Data access APIs

This section conforms to GEM1.0”11.5 Data access APIs” with following restrictions and

enhancements.

11.5.1 Broadcast transport protocol access APIs

There are two systems of broadcast transport protocol. One is the object carousel specified in ISO/IEC 13818-6. The other is the data carousel specified in ARIB STD-B24 Volume 3. All of them shall conform to GEM1.0 “Annex P: Broadcast transport protocol access” and use the org.dvb.dsmcc package as a broadcast transport protocol API. This absorbs the transport protocol differences in the low-order layers and realizes file access in all of the transmission protocols by common API.

For mapping to org.dvb.dsmcc API in the case of using the data carousel as specified in ARIB STD-B24 Volume 3, see Annex P in this standard. In addition, the return value of the java.io.File#lastModified() method for the file transmitted by data carousel shall be the resource version value in this standard “P.2.5 ObjectChangedEvent”. This means that the same value of org.dvb.dsmcc.ObjectChangeEvent#getNewVersionNumber() is returned.

If there is a file with a signature that has failed to be certified, follow the definitions in Chapter 12. Furthermore, if there is a directory with signatures that have failed to be certified, the list method of the java.io.File object that expresses that the directory returns “null” is used.

11.5.2 Return channel control APIs

In order to additionally specify the ARIB-specific return channel type and its acquisition method, the jp.or.arib.tv.net.rc.ARIBRCInterface interface that is specified in Annex R shall be added.

RCInterface objects acquired with the getInterface and getInterfaces method of org.dvb.net.rc.RCInterfaceManager have to mount this interface.

If the ARIB-specific return channel type that is not defined by org.dvb.net.rc.RCInterface is used, TYPE_OTHER is returned as a return value of the org.dvb.net.rc.RCInterface#getType() method. In this case, using the jp.or.arib.tv.net.rc.ARIBRCInterface#getARIBType() method enables acquisition of the ARIB-specific return channel type defined by the same interface.

11.6 Service information and selection APIs

11.6.1 ARIB-SI APIs

For this API, the jp.or.arib.tv.si package specified in Appendix M shall be used.

11.6.2 Service Selection APIs

This section conforms to GEM1.0 “11.6.2 Service selection API”.

11.6.3 Tuning APIs

This section conforms to GEM1.0 “11.6.3 Tuning API”. In addition, the following must be observed.

Tuning API is defined in DAVIC1.4.1 part9 “Annex H Tunig API”. However, DvbLocator and dvbNetworkBoundLocator specified in H.4 are excluded and ARIBLocator and ARIBNetworkBoundLocator of jp.or.arib.tv.net specified in this standard are replaced instead. Furthermore, org.davic.net.tuning.dvb.DvbNetworkInterfaceSIUtil is not applied.

11.6.4 Conditional access APIs

This section conforms to GEM1.0 “11.6.4 Conditional access API”.
Conditional access-relevant API is specified in Operation separately to meet the requirements of ARIB STD-B25.

11.6.5 Protocol-independent SI APIs

This section conforms to GEM1.0 “11.6.5 Protocol-independent SI API”. Note that Appendix O “Integration of the Java TV SI API” in this standard shall be referred to with regard to GEM1.0 “Annex O: Integration of the Java TV SI API” referred to in GEM1.0.

11.7 Common infrastructure APIs

This section conforms to GEM1.0 “11.7 Common infrastructure APIs”.

Note that ARIBLocator class as ARIB original locator is adopted as a subclass of ARIBLocator. The instance of ARIBLocator class shall be used as contents reference. See Chapter 14 “Relevant Matters of System Integration” of this standard for details.

11.8 Security

This section conforms to GEM1.0 “11.8 Security”. In addition, the following shall be specified.

11.8.1 Basic security

This section conforms to GEM1.0 “11.8.1 Basic security”.

11.8.2 APIs for return channel security

This section conforms to GEM1.0 “11.8.2 APIs for return channel security”.

11.8.3 Additional permissions classes

This section conforms to GEM1.0 “11.8.3 Additional permissions classes”. As a corresponding class to org.dvb.net.tuning.DvbNetworkInterfaceSIUtil,

jp.or.arib.tv.net.tuning.ARIBNetworkInterfaceSIUtil is defined in Annex N “Conditional access”.

11.8.4 General security issues

This section conforms to GEM1.0 “11.8.4 General security issues”.

11.9 Other APIs

This section conforms to GEM1.0 “11.9 Other APIs”.

11.10 Java permissions

This section conforms to GEM1.0 “11.10 Java permissions”.

11.11 Contents reference

The mapping described in this section is used between each locator defined in Table 14-1 “Accessible entity name, locator name and their textual notation” in 14.8 and the ARIB-J method. In this section, Java methods as well as the constructors are listed, which receive or return org.davic.net.Locator, javax.tv.locator.Locator and javax.media.MediaLocator as well as their subclass instances. The external format of these locators are text-style notation as defined in Table 14-1of Section 14.8. If the same method can receive more than one locator format, all formats listed in this section can be received.

If any of the below-listed methods is defined in its own specifications to check the entry by itself, only the formats listed below that are effective with the method defined in the specification can be received. Other locator formats defined in the specifications are rejected as specified in the relevant method. As for the methods that do not specify how to reject inappropriate locators, they should be terminated without exception. However, to use proper parameters for the creation of events or exceptions shall be within the scope of the receiver. Note that some methods may receive other undefined locators under this standard.

11.11.1 Transport stream

This section conforms to GEM1.0 “11.11.1 Transport stream”.

Instead of the DvbLocator class, the ARIBLocator class will be adopted as an ARIB original locator.
References to all effective locators as described in Table 14-1 of 14.8 “Accessible entity name, locator name and their textual notation” are considered with this “ARIB locator”.

Accordingly, the org.dvb.si.SITransprotStream.getDvbLocator() method among the methods listed in MHP1.0 “11.11.1 Transport stream” referred to in GEM1.0 is not used, but the

following methods shall be added.

jp.or.arib.tv.si.SITransportStream.getARIBLocator()

11.11.2 Network

This section conforms to GEM1.0 “11.11.2 Network”.

DVB Network in GEM1.0 shall refer to the effective networks described in Table 14-1 of 14.8 “Accessible entity name, locator name and their textual notation”.

11.11.3 Bouquet

This section conforms to GEM1.0 “11.11.3 Bouquet”.

11.11.4 Service

This section conforms to GEM1.0 “11.11.4 Service”. Note that the service described in this standard shall be identical to the GEM service. The following methods are also added.

jp.or.arib.tv.si.SIDatabase.retrieveSIService()

jp.or.arib.tv.si.SIDatabase.retrievePMTService()

jp.or.arib.tv.si.PMTService.getARIBLocator()

jp.or.arib.tv.si.SIBouquet.getSIServiceLocators()

jp.or.arib.tv.si.SIService.getARIBLocator()

In addition, the following methods shall be included in this standard, while they are not required in GEM1.0 “11.11.4 Service”.

Constructer of org.davic.net.ca.TuneRequestEvent

org.davic.net.ca.TuneRequestEvent.getLocator()

11.11.5 Program event

This section conforms to GEM1.0 “11.11.5 Program event”. In addition, the following method shall be added.

jp.or.arib.tv.si.SIEvent.getARIBLocator()

11.11.6 MPEG elementary stream

This section conforms to GEM1.0 “11.11.6 MPEG elementary stream”. In addition, the following methods shall be added.

jp.or.arib.tv.si.SIDatabase.retrievePMTElementaryStreams()

jp.or.arib.tv.si.PMTElementaryStream.getARIBLocator()

See 14.8 for the ARIBLocator class to be used for these methods.

11.11.7 File

This section conforms to GEM1.0 “11.11.7 File”.

For the definition of File entity or Directory entity, see Table 14-1 of 14.8 “Accessible entity name, locator name and their textual notation”.

11.11.8 Directory

This section conforms to GEM1.0 “11.11.8 Directory”. Note that GEM Locator shall be interpreted as ARIB Locator.

11.11.9 Drip feed decoder

This section conforms to GEM1.0 “11.11.9 Drip feed decoder”.

11.11.10 Irrelevant

This section conforms to GEM1.0 “11.11.10 Irrelevant”.

11.11.11 Methods working on many locator types

This section conforms to GEM1.0 “11.11.11 Methods working on many locator types”.

11.11.12 Support for the HTTP protocol in ARIB-J

This section conforms to GEM1.0 “11.11.12 Support for the HTTP Protocol in DVB-J”.

Chapter 12 Security

This Chapter 12 conforms to GEM 1.0 “12 Security,” provided that the systems specified in 12.1 below are applied for the equivalent function corresponding to “RCMM” listed in Table 15-1 of Chapter 15.

12.1 Root certificate management

12.1.1 Outline

For root certificate management in ARIB, the storage area of the root certificate inside the receiver is specified in ARIB STD-B24 Volume 2 Appendix 1 “9.4 Guidelines on Root Certificate”. Designating the root certificate storage number, the root certificate ID and the root certificate version to the storage area directs from the broadcasting station the storage of the transmitted root certificate into the receiver.

Considering conformance to the operation guideline, the format of the root certificate management message (Root Certificate Management Message), in order to store the root certificate storage number, the root certificate ID and the root certificate version, is defined in 12.1.3 in this standard in addition to the GEM requirement information. Under this standard, the root certificate stored inside the receiver can be added or removed by transmitting this root certificate management message. In addition, it is possible to transmit the root certificate using the root certificate transmission module specified in ARIB STD-B24. The root certificate descriptor stored in the module information area of DII determines in which way the certificate is handled. See 12.1.2 for details of the descriptor.

Details of the process of the root certificate management based on the root certificate management message shall conform to GEM1.0 “12.9.2 Root certificate management”. See ARIB STD-B24 Volume 2, Appendix 1 “9.4 Guidelines on Root Certificate” for details of the process of the root certificate management based on the root certificate management message.

12.1.2 Root certificate descriptor

At the time of root certificate transmission under this standard, the root certificate descriptors are placed in the DII module information. The descriptors are listed in Table 12-1.

The said descriptor shows whether the root certificate to be updated is stored in the root certificate transmission module (module_id=0xFFFF). At the same time, it restores the flag that indicates whether the root certificate management message is placed on the carousel. This descriptor is placed in order to lighten the load of the filtering process. See ARIB STD-B24 Volume 2, Appendix 1 “9.4 Guidelines on Root Certificate” for the information field operation in the root certificate descriptor in the case of using the root certificate transmission module

(module_id=0xFFFF).

Table 12-1 Root certificate descriptor in ARIB STD-B23

Data structure	Number of bits	Bit string
root_certificate_descriptor() {		
escriptor_tag	8	uimsbf
escriptor_length	8	uimsbf
oot_certificate_type	1	bslbf
cmm_distribute_flag	1	bslbf
eserved	6	bslbf
if (root_certificate_type == 0) {		
for (i=0; i<8 ; i++) {		
root_certificate_id		
root_certificate_version		
}		
} else {		
for (i=0; i<8 ; i++) {		
reserved		
}		
}		
}		

Description of Root Certificate Descriptor:

descriptor_tag (descriptor tag): In this 8-bit field, 0xCA is stored to indicate the said descriptor.

root_certificate_type (root certificate type): This 1-bit field indicates the type of the root certificate. In the case of '1', this indicates that the said root certificate is exclusive to the broadcaster. In the case of '0', the said root certificate is general purpose. (For the definition of a broadcaster-exclusive certificate and general-purpose certificate, see ARIB STD-B24 Volume 2, Appendix 1 "9.4 Guidelines on Root Certificate"). When the root certificate management message defined in 12.1.3 is transmitted, '0' is stored in the field.

rcmm_distribute_flag (RCMM distribution flag): This 1-bit field indicates whether the root certificate management message (RCMM) is placed on the carousel. In the case of '0', RCMM is not placed. In the case of '1', RCMM is placed. See 12.1.4 for details of placement on the carousel.

root_certificate_id (root certificate ID): When the value of this field is 0xFFFFFFFF, this indicates that the root certificate to be stored in the corresponding root certificate storage area

is not included in the root certificate transmission module (module_id=0xFFFF).

When the value of this field is other than 0xFFFFFFFF, this indicates that the root certificate to be stored in the corresponding root certificate storage area is included in this root certificate transmission module. In this case, the field value indicates ID that identifies the root certificate.

root_certificate_version (root certificate version number): When the corresponding root_certificate_id value is 0xFFFFFFFF, the value of this field is always 0xFFFFFFFF. In other cases, this field indicates the version number of the root certificate. Note that this is not the version number of the certificate format.

When root certificate management is conducted using only the root certificate management message, '0' shall be stored in the root_certificate_type field of this descriptor. In addition, all root_certificate_id fields and root_certificate_version fields store 0xFFFFFFFF.

12.1.3 Root certificate management message format

The format of the root certificate management message (RCMM) is coded in the form of ASN.1DER as shown below, and calculation of the digital signature is performed by this whole message. See GEM1.0 "12.11 The internet profile of X.509 (informative)" for details of the format.

```
RCMM ::= SEQUENCE {
        aribRcmm        uRCMM,
        signatures      SET OF SignatureInfo }

uRCMM ::= SEQUENCE {
        issuer                  Name,
        thisUpdate              Time,
        nextNbOfSignatures      INTEGER         OPTIONAL,
        addedCertificates       SET OF AddedCertificate,
        removedCertificates     SET OF RemovedCertificate }

AddedCertificate ::= SEQUENCE {
        storageIdentifier       CertificateStorageIdentifier,
        certificateBody         Certificate }

RemovedCertificate ::= SEQUENCE {
        storageIdentifier       CertificateStorageIdentifier,
        certificateReference    CertificateReference }
```

issuer(issuer): This is ID to specify the provider of the digital certificate that issued the message.

thisUpdate(updated day and hour): This is the date when the message was issued.
nextNbOfSignatures (number of signatures for the next): This field indicates the minimum number of effective signatures required for the RCMM message. The value takes effect from the next RCMM message transmission.

addedCertificates (additional root certificate): This is the list of root certificates to be added in the root certificate storage area. In this list, the main body of the root certificates to be added is stored.

removedCertificates (root certificate to be removed): This is the reference list of root certificates to be removed from the root certificate storage area.

storageIdentifier (storage information ID): This stores the root certificate storage number, the root certificate ID and the root certificate version.

```
CertificateStorageIdentifier ::= SEQUENCE {
        storageId                INTEGER,
        rootCertificateId        INTEGER,
        rootCertificateVersion   INTEGER }
```

storageId (root certificate storage number): This field indicates the root certificate storage number.

rootCertificateId (root certificate ID): This field indicates the ID that identifies the root certificate. For this ID, a value is uniquely assigned to the root certificate that is operated in the domestic digital broadcasting.

rootCertificateVersion (root certificate version): This field indicates the root certificate version number. Note that this is not the version number of the certificate format.

```
CertificateReference ::= SEQUENCE {
        issuerName      Name,
        serialNumber    CertificateSerialNumber }

SignatureInfo : : = SEQUENCE {
        signerName              Name,
```

```
    signatureAlgorithm      AlgorithmIdentifier,
    signatureValue          BIT STRING }
```

issuerName(certificate issuer): This is the name of the issuer of the root certificate to be removed.

serialNumber(serial number): This is the serial number of the root certificate to be removed.

12.1.4 Root certificate management message transmission

The root certificate management message (RCMM) specified in 12.1.3 shall be at least provided by the broadcasting station that uses the provider of digital certificates to certify the RCMM.

The latest RCMM is placed in the root directory on the carousel with the file name shown below.

"arib.rcmm"

The RCMM in the past is placed in the root directory on the carousel with the file name shown below.

"arib.rcmm."<x> [<x> is the text-style description in decimal integers]

The default value of <x> is "1". As one RCMM file is added, the assigned value increases by one. When an unused integer is indicated, this shows the limitation of the effective range. The oldest RCMM file will have the file name "arib.rcmm.1" and then after, other file names are assigned chronologically. Note that the contents of "arib.rcmm" and "arib.rcmm."<x> shall not be redundant.

Regardless of the transmission protocol, i.e. the data carousel or object carousel, this standard shall be observed.

Chapter 13 Graphics reference model

This chapter conforms to GEM1.0 “13 Graphics reference model” with following restrictions and enhancements.

13.1 Corresponding resolution

The section shall follow Annex G “Minimum specifications of receiver”.

For the corresponding resolution, Table 13-1 is applied.

Table 13-1 Corresponding Resolution

	16: 9 Display	4: 3 Display
HGraphicsDevice Full screen resolution	Pixel aspect ratio	Pixel aspect ratio
1920 x 1080	1 : 1	——
1280 x 720	1 : 1	——
960 x 540	1 : 1	——
720 x 480	1.18518 : 1	0.888889 : 1
The above data shall conform to the details of ARIB STD-B24 Volume 1, Part 1, “Reference Model for Data Broadcasting”.		

13.1.1 Plane resolution

Each plane of resolution specified in this standard is shown in Figure 13-1.

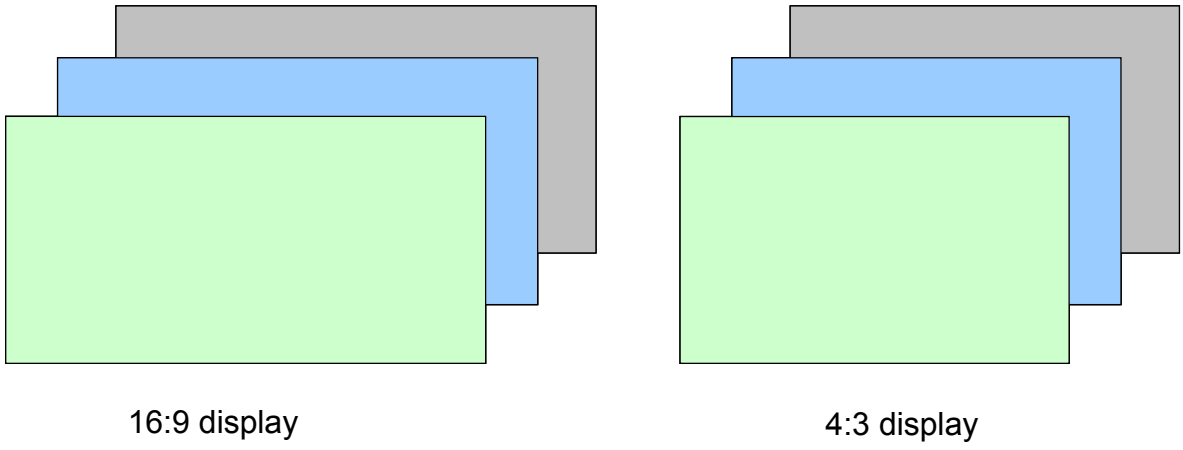


Figure 13-1 Resolution of Each Plane

13.2 Broadcast stream formats

This section conforms to GEM 1.0 “13.2 Broadcast streaming formats”. Note that the compliant broadcast stream format under this standard is only MPEG stream.

13.3 Subtitles

GEM 1.0 “13.3 Subtitles” does not require the support of API subtitles, but as mentioned in 11.4, this standard supports Subtitle control API. See “Subtitles” in Table 15-1 of Chapter 15 for details.

Chapter 14 System integration aspects

14.1 Namespace

The following format shall be used when it accesses files in the data carousel with the application in this standard.

arib-dc: //<original_network_id>.<transport_stream_id>.<service_id>[;<content_id>]
[.<event_id>]/<component_tag>/<moduleName>/<resourceName>

When it accesses the file under the directory that has a structure of multiple hierarchies, "/"shall be inserted into the string of resourceName so as to indicate the directory.

In addition, when mapping directly a resource onto a module, the following format is used. In this case, moduleName is the file name.

arib-dc: //<original_network_id>.<transport_stream_id>.<service_id>[;<content_id>]
[.<event_id>]/<component_tag>/<moduleName>

Note that the above formats are used when it specifies event messages. In this case, the event object name is specified in resourceName and 0x6 is specified for TypeValue in resourceTypeValue() in the resource media type specified in "B.2.1.2 Definitions of Resource Information in ARIB-J Application Transmission Carousel".

14.2 Reserved names

In order to avoid redundancy for the application compliant with the revised standard in the future, file names that start with arib. are specified as reserved names and are not to be used in the application.

14.3 XML Notation

This section conforms to GEM1.0 "14.3 XML notation".

14.4 Network signaling

This section conforms to GEM1.0 "14.4 Network signaling".

14.5 Text encoding of application identifiers

This section conforms to GEM1.0 "14.5 Text encoding of application identifiers".

14.6 Reserved names for persistent storage

This section conforms to GEM1.0 "14.6 Reserved names for persistent storage". Note that '~' is

also specified as a reserved word in this standard.

14.7 File and file names

This section conforms to GEM1.0 “14.7 File and file names.”. In addition, the following extensions and changes are specified.

In ARIB-J application, the root directory to be mounted as a service domain shall be designated in the following format.

arib-dc: //<original_network_id>.<transport_stream_id>.<service_id>[;<content_id>][.<event _id>]/<component_tag>

If the file name is included in the URL, conversion to expression of “file: //” shall not be performed.

14.8 Reference to locator and contents

In this standard, the ARIBLocator class is adopted as an ARIB original locator. This class is a subclass of org.davic.net.Locator. In addition, ARIBNetworkBoundLocator that inherits the ARIBLocator class is adopted as a class that corresponds to DvbNetworkBoundLocator.

The relation of the inheritance of the newly adopted classes are shown below.

class org.davic.net.Locator
implements javax.tv.locator.Locator
    |
    +- **class jp.or.arib.tv.net.ARIBLocator** [New]
        |
        +- **class jp.or.arib.tv.net.ARIBNetworkBoundLocator** [New]
            implements org.davic.net.TransportDependentLocator

Table 14-1 shows the entity types and corresponding name spaces that are available for reference with the ARIB locator.

Table 14-1 Accessible entity name, locators and other textual notation

Entity name	Textual notation
Transport stream	arib: //<original_network_id>.<transport_stream_id>
Network	Not particularly specified.
Service	arib: //<original_network_id>.<transport_stream_id>.<service_id>
Service domain	arib-dc: //<original_network_id>.<transport_stream_id>.<service_id> [;<content_id>][.<event_id>]/<component_tag>
Program event	arib: //<original_network_id>.<transport_stream_id>.<service_id>.<event_id>
MPEG elementary stream	arib: //<original_network_id>.<transport_stream_id>.<service_id>[;<content_ id>][.<event_id>]/<component_tag>[;<channel_id>]{&<component_tag> [;<channel_id>]}
File	The URL of "file:", "http:" and "https:" can be used. (Note that the conversion to the mark "file:" can be performed only when the transport protocol in the name space is "dvb:".) The URL that shows the file under the Service Domain is as follows. arib-dc: //<original_network_id>.<transport_stream_id>.<service_id>[;< content_id>][.<event_id>]/<component_tag>/<moduleName>[/<resourceName >] resourceName is stored in the resource name inside ExtendedResourceInfo() as specified in Table B.3 of Annex B and is shown as follows. resourceName = *(<directoryName>/)<fileName> When one resource does mapping directly onto a module, <moduleName> is the file name. Note that <moduleName> is stored in the name descriptor of ModuleInfoByte inside DII.
Directory	The URL of "file:", "http:" and "https:" can be used. (Note that conversion to expression of "file:" can be performed only when the transport protocol in the name space is "dvb:"). The URL that shows the file under the Service Domain is as follows. arib-dc: //<original_network_id>.<transport_stream_id>.<service_id>[;< content_id>][.<event_id>]/<component_tag>/<moduleName>[*(/< directoryName>)] The directoryName is stored as a part of the resource name in ExtendedResourceInfo() that is specified in Table B.3 of Annex B and it is shown as follows. resourceName = *(<directoryName>/)<fileName>
Drip feed decoder	"dripfeed: //"
MPEG-I frame to be transmitted by a still picture carousel	arib-ic: //…[<event_id>]/<component_tag>/<I-frame_ID>
Built-in sound in receiver	To be specified by operational guideline.

The usable letters for <resourceName> in BNF notation are as follows.

resourceName	= path_segments
directoryName	= segment
fileName	= segment
path_segments	= segment * (“/” segment)
segment	= 1*pchar
pchar	= unreserved | escaped |
	“: “ | “@” | “&” | “=“ | “+” | “$” | “,”
unreserved	= alphanum | mark
mark	= “-” | “_” | .”“ | “!” | “~” | “*” | ““ | “(“ | “)”
escaped	= “%” hex hex
hex	= digit | “A” | “B” | “C” | “D” | “E” | “F” |
	“a” | “b” | “c” | “d” | “e” | “f”

Each name is case-sensitive and recognized as equivalent even if encoding with % hex hex is performed (i.e. coding in the form “%a0%ff”).

Each ID such as original_network_id is displayed in hexadecimal strings. However, in this case, it doesn't include hexadecimal symbols such as "0x" at the start of the string or "h" at the end of the string. Instead, "0" is assigned at the beginning of the string as needed in number so as to make the string an appropriate fixed-length string for the number of bits. Furthermore, the hexadecimal character is not case sensitive.

14.9 Service ID

In Java TV, two types of locator, one relying on the transmission protocols and the other not relying on the transmission protocols, are specified as a locator for service identification. This standard applies the one that relies on the transmission protocols.

14.10 CA System

This standard shall conform to the conditional access system specified in ARIB STD-B25.

Chapter 15 Detailed platform profile definitions

This chapter specifies the definition of profile and each monomedium, protocol and API to be operated by the profile.

This chapter conforms to GEM1.0 “15 Detailed platform profile definitions”.

According to the GEM1.0 “15.6 Functional equivalents,” the functional equivalent table is provided in Table 15-1. The “Name”s in the table correspond to the “Name”s of Table 7 of GEM 1.0 “15.6 Functional equivalents”.

Table 15-1 Functional Equivalents

Name	Chapter in GEM	Implementation in B23	Remarks
Arch	5, “Basic Architecture”	Not specified	The basic architecture is specified by operation guideline.
Carousel	6.2.5, “Object carousel” See also 11.7.2, “Application discovery and launching APIs”	Annex B (Data carousel) MHP Annex B, ISO/IEC 13818-6 (DSM-CC Object carousel)	See ARIB STD-B24 Volume 3 for Annex B. In the case of using a data carousel, Annex B of this standard is applied. In the case of using an object carousel, MHP Annex B is applied. Note that transport_stream_id, original_network_id, service_id of dvb_service_location() in Table B.26: DVB Carousel NSAP Address shall follow the semantics of ARIB-SI. In this standard, either protocol of carousel is selected.
IP MPE	6.2.6, “Protocol for delivery of IP multicast over the broadcast channel”	Not specified	Outside the scope of this standard
SI	6.2.9, “Service information”	Part 2, 6.11	See ARIB STD-B10 for 6.11.
	11.6.1, “Signalling-bound service information API”	Part 2, Appendix M (jp.or.arib.tv.si package)	

	Annex O, "Integration of the JavaTV SI API"	Part 2, Appendix O	
Broadcasting IP signalling	6.2.10, "IP signalling"	Not specified	Outside the scope of this standard
Audio	7.2.1, "Audio"	Part 1, 4.1 MPEG-2 Audio Part 1, 4.2 PCM (AIFF-C) Part 1, 4.3 MPEG-4 Audio Part 1, 4.4 Coding of additional sound	Including the details of MHP 7.2.1.
Video	7.2.2, "Video"	Part 1, 2.1 MPEG-1 Video Part 1, 2.2 MPEG-2 Video Part 1, 2.3 MPEG-4 Video	Including the details of MHP 7.2.2.
Subtitles	7.2.3, "Subtitles"	Part 1, Chapter 7	
	11.4, "Presentation APIs," classes related to subtitles	MHP 11.4.2.5.1 MHP 11.4.2.5.2	As for the coding of subtitles as monomedia, this standard employs the methods specified in ARIB STD-B24, not that of DVB. (See ARIB STD-B24 of Part 1, Chapter 7 of this standard) However, for the class definition in 11.4.1, MHP's standards, which is excluded from the standard in GEM, is adopted again. This policy is the same as the API policy of DSM-CC for a data carousel.
Application signaling	10.2, "Program specific information"	Part 2, Chapter 10	See ARIB STD-B10 of Chapter 10.
	10.4, "Application description" 10.5, "DVB-J specific application description"	Part 2, Chapter 10	
	11.7.2, "Application discovery and launching APIs and bindings"	GEM	
Conditional Access	11.4, "Presentation APIs," classes related to conditional access 11.6.4, "Conditional access API"	Part 2, 11.4.3 Part 2, 11.6.4	

Content Referencing	11.7.6, “Content referencing”	Part 2, 11.7	This standard adopts ARIBLocator as the subclass of org.davic.net.Locator. ARIBLocator is equivalent to DvbLocator in MHP.
	11.11.11, “Methods working on many locator types”	GEM	
	14.1, “Namespace mapping”	Part 2, 14.1	The definition of name space in ARIB shall be functionally equivalent to the name space mapping of MHP as defined in MHP 14.1.
	14.9, “Service identification”	Part 2, 14.9	This standard supports only the locator that relies on the transmission channel. This is compliant with GEM.
Graphics Resolution	D.1, “Horizontal resolution”	Part 2, Annex N	See STD-B21.
	G.1.1, “Device resolution for Standard Definition”	GEM	
	G.4, “Resident fonts and text rendering”	GEM	
Text Wrapping	D.2, “Text wrapping setting is true”	GEM	
Minimum CLUT	G.1.2, “Minimum Colour Lookup Table”	MHP G.1.5	
RCMM	12.9.2, “Root Certificate Management”	Part 2, Chapter 12	

Chapter 16 Registry of constants

This chapter defines the constants and their values used in ARIB-J.

This chapter also conforms to GEM1.0 “16 Registry of constants”. The values of the constants defined in ARIB-J-specific API are as follows.

jp.or.arib.tv.net.rc.ARIBRCInterface		
public static final int	ARIB_TYPE_ETHERNET_DHCP	403
public static final int	ARIB_TYPE_ETHERNET_FIXED_IP	402
public static final int	ARIB_TYPE_ETHERNET_PPPOE	401
public static final int	ARIB_TYPE_MOBILE_PHONE	300
public static final int	ARIB_TYPE_MOBILE_PHONE_CDMA_CELLUARSYSTEM	305
public static final int	ARIB_TYPE_MOBILE_PHONE_DS_CDMA	303
public static final int	ARIB_TYPE_MOBILE_PHONE_MC_CDMA	304
public static final int	ARIB_TYPE_MOBILE_PHONE_PDC	301
public static final int	ARIB_TYPE_MOBILE_PHONE_PDCP	302
public static final int	ARIB_TYPE_PHS	200
public static final int	ARIB_TYPE_PHS_PIAFS20	201
public static final int	ARIB_TYPE_PHS_PIAFS21	202
jp.or.arib.tv.si.DescriptorTag		
public static final short	AUDIO_COMPONENT	196
public static final short	BASIC_LOCAL_EVENT	208
public static final short	BOARD_INFORMATION	219
public static final short	BOUQUET_NAME	71
public static final short	BROADCASTER_NAME	216
public static final short	CA_CONTRACT_INFO	203
public static final short	CA_EMM_TS	202
public static final short	CA_IDENTIFIER	83
public static final short	CA_SERVICE	204
public static final short	CABLE_DELIVERY_SYSTEM	68
public static final short	CAROUSEL_COMPATIBLE_COMPOSITE	247
public static final short	COMPONENT	80
public static final short	COMPONENT_GROUP	217
public static final short	CONNECTED_TRANSMISSION	221
public static final short	CONTENT	84
public static final short	CONTENT_AVAILABILITY	222

public static final short	COUNTRY_AVAILABILITY	73
public static final short	DATA_COMPONENT	253
public static final short	DATA_CONTENTS	199
public static final short	DIGITAL_COPY_CONTROL	193
public static final short	DOWNLOAD_CONTENT	201
public static final short	EMERGENCY_INFORMATION	252
public static final short	EVENT_GROUP	214
public static final short	EXTENDED_BROADCASTER	206
public static final short	EXTENDED_EVENT	78
public static final short	HIERARCHICAL_TRANSMISSION	192
public static final short	HYPER_LINK	197
public static final short	LDT_LINKAGE	220
public static final short	LINKAGE	74
public static final short	LOCAL_TIME_OFFSET	88
public static final short	LOGO_TRANSMISSION	207
public static final short	MOSAIC	81
public static final short	NETWORK_IDENTIFICATION	194
public static final short	NETWORK_NAME	64
public static final short	NODE_RELATION	210
public static final short	NVOD_REFERENCE	75
public static final short	PARENTAL_RATING	85
public static final short	PARTIAL_RECEPTION	251
public static final short	PARTIAL_TRANSPORT_STREAM	99
public static final short	PARTIALTS_TIME	195
public static final short	REFERENCE	209
public static final short	SATELLITE_DELIVERY_SYSTEM	67
public static final short	SERIES	213
public static final short	SERVICE	72
public static final short	SERVICE_LIST	65
public static final short	SHORT_EVENT	77
public static final short	SHORT_NODE_INFORMATION	211
public static final short	SI_PARAMETER	215
public static final short	SI_PRIME_TS	218
public static final short	STC_REFERENCE	212
public static final short	STREAM_IDENTIFIER	82
public static final short	STUFFING	66
public static final short	SYSTEM_MANAGEMENT	254

public static final short	TARGET_AREA	198
public static final short	TERRESTRIAL_DELIVERY_SYSTEM	250
public static final short	TIME_SHIFTED_EVENT	79
public static final short	TIME_SHIFTED_SERVICE	76
public static final short	TS_INFORMATION	205
public static final short	VIDEO_DECODE_CONTROL	200
jp.or.arib.tv.si.PMTStreamType		
public static final byte	DSMCC_DATA_CAROUSEL	13
public static final byte	INDEPENDENT_PES	6
public static final byte	MPEG1_AUDIO	3
public static final byte	MPEG1_VIDEO	1
public static final byte	MPEG2_AAC_AUDIO	15
public static final byte	MPEG2_AUDIO	4
public static final byte	MPEG2_VIDEO	2
public static final byte	MPEG4_VIDEO	16
public static final byte	MPEG4_AVC_VIDEO	27
jp.or.arib.tv.si.SIDatabase		
public static final int	RETRIEVE_ALL_INFORMATIONS	-1
public static final int	RETRIEVE_CURRENT_SELECTED	-2
jp.or.arib.tv.si.SIInformation		
public static final short	FROM_CACHE_ONLY	0
public static final short	FROM_CACHE_OR_STREAM	1
public static final short	FROM_STREAM_ONLY	2
jp.or.arib.tv.si.SIMonitoringType		
public static final byte	BOUQUET	2
public static final byte	BROADCASTER	7
public static final byte	NETWORK	1
public static final byte	PMT_SERVICE	4
public static final byte	PRESENT_FOLLOWING_EVENT	5
public static final byte	SCHEDULED_EVENT	6
public static final byte	SERVICE	3
jp.or.arib.tv.si.SIRunningStatus		
public static final byte	NOT_RUNNING	1
public static final byte	PAUSING	3
public static final byte	RUNNING	4
public static final byte	STARTS_IN_A_FEW_SECONDS	2
public static final byte	UNDEFINED	0

jp.or.arib.tv.si.SIServiceType		
public static final short	BOOKMARK_LIST	170
public static final short	DATA	192
public static final short	DATA_EXCLUSIVE_FOR_ACCUMULATION	169
public static final short	DATA_FOR_ACCUMULATION_IN_ADVANCE	168
public static final short	DIGITAL_AUDIO	2
public static final short	DIGITAL_TELEVISION	1
public static final short	ENGINEERING_DOWNLOAD	164
public static final short	PROMOTION_DATA	167
public static final short	PROMOTION_SOUND	166
public static final short	PROMOTION_VIDEO	165
public static final short	SPECIAL_AUDIO	162
public static final short	SPECIAL_DATA	163
public static final short	SPECIAL_VIDEO	161
public static final short	UNKNOWN	-1

Annex A External reference, errata, clarifications and exemptions

This Annex conforms to GEM1.0 “Annex A: External references; errata, clarifications and exemptions”.

Annex B Broadcast filesystem and trigger transport

BML contents in ARIB STD-B24 are transmitted based on the data carousel. The data are transmitted using DownloadDataBlock message and DownloadInfoIndication message among the User-to-Network download protocols of DSM-CC data carousel specifications specified in ISO/IEC 13818-6.

This Annex describes additional definitions that are necessary when data are transmitted based on the ARIB data carousel as defined in ARIB STD-B24 Volume 3. Accordingly, standard other than those described shall conform to ARIB STD-B24 Volume 3.

See Chapter 15, Table 15-1 “Carousel” for the definitions of transmission based on the object carousel.

B.1 Service domain

This section conforms to the requirements of GEM1.0 “B.1 Service domain” with following restrictions and enhanments to meet the requirements mentioned above when adopting the data carousel.

B.1.1 Root directory definition of carousel

In ARIB-AE specifications, the following is defined as the format of the service domain and shall be mounted on each carousel as a root directory.

arib-dc: //<original_network_id>.<transport_stream_id>.<service_id>[;<content_id>]
[.<event_id>]/<component_tag>

If the above is mounted as a URL to be the ARIB Locator, using this as a root directory, the ARIB-J application can access the file (object) in the carousel via the relative path. See 14.1 “Name Space” for the name space in the ARIBLocator.

In addition, if there is an access that crosses over more than two data carousels, reference shall be made by switching the service domain.

B.1.1.1 NSAP address

In ARIB, carousel_id is not defined. DownloadId of DII is stored and used instead.

Table B-1 NSAP Address Space in ARIB

Field	AFI	Type	CarouselI	SpecifierT	Specifier	Transport	Original_	Service_i	reserved

			d	ype	Data	_stream_id	Network_id	d	
Number of bits	8	8	32	8	24	16	16	16	32
Value	0x00	0x00	+	0x01	0x819292 [T.B.D]	+	+	+	0xFFFFFFFF
	=Private Use	=Object Carousel NSAP	=DownloadId	=IEEE OUI	=ARIB OUI (Conts.)				

An identification Value to specify “ARIB” shall be defined for SpecifierData. The ID value shall be acquired in consultation with IEEE.

In the CarouselId field, as shown above, the DownloadId of DII shall be stored. This standard regards DownloadId as CarouselId.

When the DownloadId of the data carousel being transmitted is changed (e.g. change of data_event_id), the currently mounted service domain shall be unmounted (equivalent to the detach operation defined in DSM-CC). See the Annex P “Broadcast transport protocol access” for details.

B.2 Requirements of file system

B.2.1 Static requirements

“Table B.1: File system signaling requirements” of GEM1.0 “B.2.1 Static requirements” shall be satisfied. The following definitions shall be added for ARIB STD-B24 Volume 3 “6 Data Carousel Transmission Protocol” in order to satisfy the requirements.

B.2.1.1 Definitions for resource list in ARIB-J application transmission carousel

In order to satisfy the requirements for the file system required in GEM1.0 “B.2 File system requirements,” the following resource list is defined for the ARIB-J application transmission carousel. Media type (Content-Type) is “application/X-arib-ExtendedResourceList” so as to differentiate it from the resource list of the ARIB STD-B24.

Table B-2 Coding of resource list for ARIB-J (X-arib-ExtendedResourceList)

Syntax	Bits	type

X-arib-ExtendedResourceList {
num_of_resources
For (i=0; i< num_of_resources; i++) {
ExtendedResourceInfo()
}
}
	16	uimsbf

Description of X-arib-ExtendedResourceList:
num_of_resources (number of resources): This 16-bit field indicates the number of resources to be included in the identical module. This also includes the resources for the event message description and stream description. Note that the list itself is not included.

ExtendedResourceInfo () (extended resource information): The data structure stores information for one resource in the module. This is specified in Table B-3.

B.2.1.2 Definitions of resource information for ARIB-J application transmission carousel
The following resource information is defined for the carousel of the ARIB-J application transmission so that the access to the information such as event message description and stream description is available from the carousel.

If reference for an event message or stream data is necessary by object name via carousel, the said resource information is also stored on the said carousel as in the same way of the file on the carousel.

As for the file type that indicates the event message, 0x6 (Type Value) is defined.

Table 3 Coding of ExtendedResourceInfo ()

Syntax	Bits	Type
ExtendedResourceInfo(){		
ResourceInfoLength	16	uimsbf
ResourceOffset	32	uimsbf
HeaderLength	16	uimsbf
ResourceLength	32	uimsbf
ResourceTypeValue() {		
TypeValue	4	bslbf
SubtypeValue	12	bslbf
}		
ResourceNameLength	16	uimsbf
for (j=0; j< resourceNameLength; j++) {		
text_char	8	uimsbf
}		
stream_flag	1	bslbf
audio_flag	1	bslbf
video_flag	1	bslbf
data_flag	1	bslbf
reserved	4	uimsbf
if (stream_flag ==0x1) {		
component_tag	8	uimsbf
npt_component_tag	8	uimsbf
dsm_contented	7	uimsbf
stream_event_flag	1	bslbf
duration	32	uimsbf
if (stream_event_flag == 0x1)		
reserved	4	uimsbf
event_msg_group_id	12	uimsbf
event_msg_type	8	uimsbf
eventIds_count	8	uimsbf
for (j=0 ; j< eventIds_count; j++) {		
event_msg_id	16	uimsbf
eventName_length	8	uimsbf
for (k=0; k< eventName_length;k++) {		
eventName_data	8	uimsbf
}		
}		
}		
} else {		
for (k=0; k<N; k++){		
reserved	8	uimsbf
}		
}		
}		
}		

Description of ExtendedResourceInfo ():

ResourceInfoLength (resource information length): This 16-bit field indicates the number of bytes immediately after the resource information field to the end of the resource information.

ResourceOffset (resource offset): This 32-bit field indicates the start byte of the body-part of the

resource to be indicated by the resource information. This value is offset from beginining of the module.

HeaderLength (header length): This 16-bit field indicates the number of bytes of the header space in the body-part of the resource to be indicated by the resource information. This headerLength does not include CRLF length (2 bytes) as a delimiter to the entity-body.

ResourceLength (resource length): This 32-bit field indicates the length of the resource. The value assigned to this field coincides with the Content-Length field value in the body-part header space. Note that the value is "0" if the resource information makes reference to the ES of the event message description or stream description and so on because contents files are absent on the carousel.

ResourceTypeValue () (resource media type): The data structure is 16-bit. It indicates the media type of the resource indicated by the said resource information. The details of data structure shall conform to ARIB STD-B24 Volume 2 "9.1.2.3 Resource List for Multipart Format Modules". Note that 0x6 (Type Value) is defined for the file type that indicates the event message.

ResourceNameLength (resource name length): This 16-bit field indicates the byte length of the subsequent resource name.

text_char (resource name): This is an 8-bit character information field. A directory name can be included to store with the symbol"/," which is recognized as a usable character. The resource name designated by this field has the same value if it is stored in the Content-Location of the body-part header space.

stream_flag (stream flag): This indicates that the said resource is the resource information for the stream. When it is the resource information for the stream, '1' is stored. When it is the resource information for the event message, '1' is stored as well. In this case, the file corresponding to the said resource on the data carousel is not transmitted.

audio_flag (audio stream flag): If the said resource is the resource information for the stream, '1' is stored when the stream is an audio stream. This is equivalent to the audio field in BIOP: : StreamMessage on the object carousel.

video_flag (video stream flag): If the said resource is the resource information for the stream, '1' is stored when the stream is a video stream. This is equivalent to the video field in BIOP: : StreamMessage on the object carousel.

data_flag (data stream flag): If the said resource is the resource information for the stream, '1' is

stored when the stream is a data stream. This is equivalent to the data field in BIOP: : StreamMessage on the object carousel.

component_tag (resource reference component tag): This is a component tag that indicates the ES in which the said resource stream is being transmitted.

npt_component_tag (NPT reference component tag): This is a component tag that indicates the ES in which the NPT reference descriptor to which the stream of the said resource should refer is being transmitted.

dsm_contentId: This 7-bit field stores the identifier to specify the NPT reference descriptor to which the stream of the said resource should refer.

stream_event_flag (stream event flag): If the steam of the said resource is the one that transmits the event message, '1' is stored.

Duration (duration): This 32-bit field indicates the duration of the stream of the said resource. If the duration cannot be defined, '0' is stored.

event_msg_group_id (message group ID): This 12-bit field indicates the identifier that identifies the message group the application should receive.

event_msg_type (message type): This is an identifier to indicate the type of the message.

eventIds_coun t(number of events): This indicates the number of messages of the message group that are identified by this resource name.

event_msg_id (message ID): This 16-bit field indicates the identifier that identifies each event message. This is equivalent to EventId of org.dvb.dsmcc API.

eventName_length (event name length): This indicates the length of the identification name of each event message. This event name is used for org.dvb.dsmcc API.

eventName_data (event name data): The identification name that identifies each event message is stored.

Based on the definitions specified above, mapping of each object of the BIOP messages in DVB object carousel shall be carried out on the ARIB data carousel as follows.

Table B-4 BIOP object mapping of message

	MHP (Object Carousel)	ARIB (Data Carousel)
File	BIOP: FileMessage	Equivalent to resource file
Directory	BIOP: DirectoryMessage	Equivalent to module name, resource name. The module name is stored in Name descriptor of DII module information. The resource name is stored in ExtendedResourceInfo ().
Stream	BIOP: StreamMessage	Equivalent to ExtendedResourceInfo (). The resource name is equivalent to the object name. In this case, TypeValue is 0x3 in case of video stream and 0x4 in case of audio stream.
StreamEvent	BIOP: StreamEventMessage	Equivalent to ExtendedResourceInfo (). The resource name is equivalent to the object name. In this case, TypeValue is 0x6.

B.2.1.3 Caching behavior

This section conforms to GEM1.0 “B.2.1.1 Caching behavior”. The caching priority descriptor is added to DII module information so as to transmit the information relevant to caching behavior required in B.2.1.1 of GEM1.0. The structure of the descriptor is shown below.

Table B-5 Syntax of Caching priority descriptor

Syntax	Bits	type	Comment
caching_priority_descriptor() {			
descriptor_tag	8	uimsbf	0x71
descriptor_length	8	uimsbf	
priority_value	8	uimsbf	
transparency_level	8	uimsbf	
}			

Description of caching_priority_descriptor ():

descriptor_tag (descriptor tag): This indicates the caching priority descriptor and assigns 0x7.

priority_value (priority value): This indicates the priority value of the caching. The indicated caching priority is the one of the module where the descriptor is stored as module information. The higher the value is, the more important the caching is.

transparency_level (transparency level): This indicates the transparency level to be used at the terminal where object caching is performed. The assignable values are as follows.

Table B-6 Range of transparency level

Value	Description
0	reserved

1	Transparent caching
2	Semi-transparent caching
3	Static caching
4...255	reserved for future use

If the descriptor is not stored, the following default values are assigned.

- priority_value = 128
- transparency_level = 1 (Transparent caching)

B.2.2 Update of file system

In the ARIB data carousel, the version value can be assigned to each module. And mapping to acquire the value via org.dvb.dsmcc API enables the file system update information to be detected. See the Annex P “Access method to transport protocol” for details.

B.2.3 Unavailability of data carousel

The term “unavailability” in this section means the state that the carousel becomes unavailable permanently.

The data carousel is defined as follows. See this section when the data carousel is used.

When the ES component of the data carousel that transmits the ARIBJ application from the PMT is out of use.

When the contents of the data coding descriptor for the ES component of the data carousel that transmits the ARIBJ from the PMT are changed.

When the download_id that is associated with the data carousel is changed.

When programs are eliminated from the PAT

When a certain time period has passed and it is judged that signals can no longer be received (the time period depends on the mounting status).

B.3 Description for stream

The information described in Table B.3: Stream description can be acquired in the way described below in the ARIB data carousel, according to the requirement of GEM 1.0 “B.3 Stream description”.

Table B-7 Acquisition of descriptor for stream

Function	Description
npt_source	The value of the npt_component_tag in Table B-3 ExtendedResourceInfo () enables specification of where to refer.
stream_locator	The reference component tag can be obtained by ExtendedResourceInfo () conponent_tag value.
duration	Obtainable by ExtendedResourceInfo ()

audio_stream	Identifiable by audio_flag of ExtendedResourceInfo (). (The detailed format can be specified by ResourceTypeValue ()).
video_stream	Identifiable by video_flag of ExtendedResourceInfo () (The detailed format can be identified by ResourceTypeValue ())
data_stream	Identifiable by data_flag of ExtendedResourceInfo () (The detailed format can be identified by ResourceTypeValue ())
is_mpeg_program	Discriminable by the stream form ID of the stream for reference.

B.4 Trigger signaling

In this standard, “Trigger defined in GEM 1.0 “B.4 Trigger signaling“ is equivalent to the event message specified in ARIB STD-B24 Volume 3 “Chapter 7 Event Message Transmission Protocol”.

B.4.1 Trigger object

In this standard, the equivalent information to the trigger object defined in GEM 1.0 “B.4.1Trigger object” shall be transmitted by the extended resource information.

The resource name in the resource information corresponds to the trigger object name and the information of each trigger is stored in the corresponding resource information. The ES that transmits the trigger can be referred by the component tag stored in the resource information.

B.4.2 Trigger event

The function of a trigger event defined in GEM 1.0 “B 4.2 Trigger event” shall be specified as follows in this standard.

Table B-8 Corresponding function of trigger event

Function		Description
event_id	14bits	Equivalent to event_msg_id (16bit)
is_do_it_now	boolean	Equivalent to the case ‘time_mode=0x00’
mpeg_npt	33bits	To be calculated based on MPT reference descriptor in case of “time_mode=0x02”.
pay_load	N bytes	Equivalent to private_data_byte of general-purpose event message descriptor.

Not transmitted as a flag that corresponds to “is_do_it_now” in the data carousel but this can be discriminated by the time_mode field value in General_event_descriptor () that transmits event messages. In the ARIB, time_mode=0x00 is equivalent to immediate firing mode.

B.4.2.1 Extrapolation of NPT values

This section shall follow the requirements in GEM1.0 “B.4.2.1 Extrapolation of NPT values”.

B.4.2.2 Monitoring of trigger events

This section shall follow the requirements in GEM1.0 “B.4.2.2 Monitoring of trigger events”.

Annex C References

The relevant documents such as standards for an Application Execution Engine Platforms with this standard are as follows.

"Standard Television Data Multiplex Broadcasting by Transmission Method Using Vertical Blanking Interval" ARIB STD-B5

"Service Information for Digital Broadcasting System" ARIB STD-B10

"Receiver for Digital Broadcasting" ARIB STD-B21

"Data Coding and Transmission Specification for Digital Broadcasting" ARIB STD-B24

"Conditional Access System Specifications for Digital Broadcasting" ARIB STD-B25

In addition, GEM 1.0 "Annex C: References" shall be refered.

Annex D Text presentation

D.1 Alphanumeric fonts

Text presentation of alphanumeric fonts on the screen shall conform to GEM 1.0 “Annex D: Text presentation”.

D.2 Japanese character fonts

Text presentation of Japanese character fonts on the screen is specified in operational guidline separately. The on-screen text presentation of alphanumeric characters included in Japanese character fonts is not necessarily to follow the definition of D.1.

D.3 Other fonts

Text presentation of fonts other than those specified in D.1 and D.2 is specified in operational guideline separately.

Annex E Character set

The minimum character set of this standard shall include all characters specified below.

E.1 Latin alphabet

This section conforms to GEM1.0 “Annex E: Character set”.

E.2 Basic characters

The structure of basic characters is as follows.

- The following three collections among the seven collections described in JIS X 0221-1-2001 Appendix 1

− BASIC JAPANESE (Basic Japanese character collection)
− FULLWIDTH ALPHANUMERIC (Fullwidth alphanumeric character collection for compatibility)
− HALFWIDTH KATAKANA (Halfwidth Katakana character collection for compatibility)

Note)
Characters of 0x0020~0x007E except 0x0060 (GRAVE ACCENT) among BASIC JAPANESE, 0x00A3 (POUND SIGN), 0x00A5 (YEN SIGN), 0x00B0 (DEGREE SIGN), 0x00D7 (MULTIPLICATION SIGN), 0x00F7 (DIVISION SIGN), 0x2190 (LEFTWARDS ARROW), 0x2191 (UPWARDS ARROW), 0x2192 (RIGHTWARDS ARROW), 0x2193 (DOWNWARDS ARROW) and 0x221E (INFINITY) overlap with Latin alphabet.

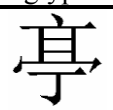
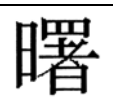
- The characters which have J source references in ISO/IEC 10646-1: 2000 “27 CJK Unified ideographs” and not included in BASIC JAPANESE.

As for CJK unified ideographs (Kanji), only J sourced glyphs shall be referred. Other glyphs of CJK unified ideographs shall not be referred to.

E.3 Basic additional kanji

The kanji listed in Table E-1 shall be added in PUA (Private Use Area).
Note that the code position of the following kanji characters is subject to change in the future.
- A character that is described as a variant in the remarks column.
- A character of which the original code position is specified in ISO/IEC 10646-2: 2001 in the original code position column (the code representation in hexadecimal value is greater than 0xFFFF).

Tabel E-1 Additional kanji characters to PUA

Example glyph	Assigned position	Original code position	Remarks
	0xE016	0x20158	
	0xE017	0x20BB7	
	0xE000		
	0xE006	0x66D9	This glyph appears as K sourced one and variant of G, T and J sourced ones.
	0xE001		
	0xE007	0x6ADB	This glyph appears as K sourced oneand variant of G, T and J sourced ones.
	0xE018	0x233CC	
	0xE019	0x233FE	
	0xE01A	0x235C4	
	0xE008	0x6D77	This glyph appears as G sourced one and variant of T, J, K and V sourced ones.
	0xE009	0x6E1A	This glyph appears as K sourced one and variant of G, T, J, and V sourced ones.
	0xE002		
	0xE01B	0x242EE	
	0xE00A	0x7422	This glyph appears as T sourced one and variant of G, J, K and V sourced ones.
	0xE00B	0x7947	This glyph appears as K sourced one and variant of G, T, and J sourced ones.

禮	0xE00C	0x79AE	This glyph appears as G sourced one and variant of T, J, K and V sourced ones.
祓	0xE003		
袂	0xE00D	0x4103	This glyph appears as T sourced one and variant of J and K sourced ones.
襃	0xE004		
舘	0xE005		
葛	0xE00E	0x845B	This glyph appears as G sourced one and variant of T, J and K sourced ones.
蓬	0xE00F	0x84EC	This glyph appears as K sourced one and variant of G, T, J, and V sourced ones.
蝕	0xE010	0x8755	This glyph appears as K sourced one and variant of G, T, and J sourced ones.
角	0xE011	0x89D2	This glyph appears as G sourced one and variant of T, J, K and V sourced ones.
辻	0xE012	0x8FBB	This glyph appears as K sourced one and variant of G, T, and J sourced ones.
鄭	0xE013	0x912D	This glyph appears as K sourced one and variant of G, T, J, and V sourced ones.
餃	0xE014	0x9903	This glyph appears as G sourced one and variant of T, J and K sourced ones.
鯖	0xE015	0x9BD6	This glyph appears as K sourced one and variant of G, T, and J sourced ones.

In addition, the characters listed in Table E-2 shall be added at their listed code postions. Glyphs of these characters shall be G sourced ones unless otherwise specified in the remarks column.

Table E-2 Kanji characters to be added by code position

Code position	Remarks
0x5307	
0x5BEC	
0x5FB7	
0x6DF8	
0x73A8	
0x7575	
0x7B7F	
0x87EC	
0x9592	
0x9AD9	
0xFA10	A variant of 0x585A
0xFA11	
0x0302	Combining characters.
0x0305	Combining characters.
0x0308	Combining characters.
0x0332	Combining characters.
0x0340	Combining characters.
0x0341	Combining characters.
0x20DD	Combining characters.

E.4 Additional character for broadcasting

The additional characters for broadcasting shown in Table E-3 shall be added. The code positions listed in Table E-4 shall be used for these characters. Note that characters with a code position in PUA are additional characters to ISO/IEC 10646-1: 2000 and ISO/IEC 10646-1: 2000/Amd.1: 2002 by this standard.

Table E-3 Additional characters for broadcasting

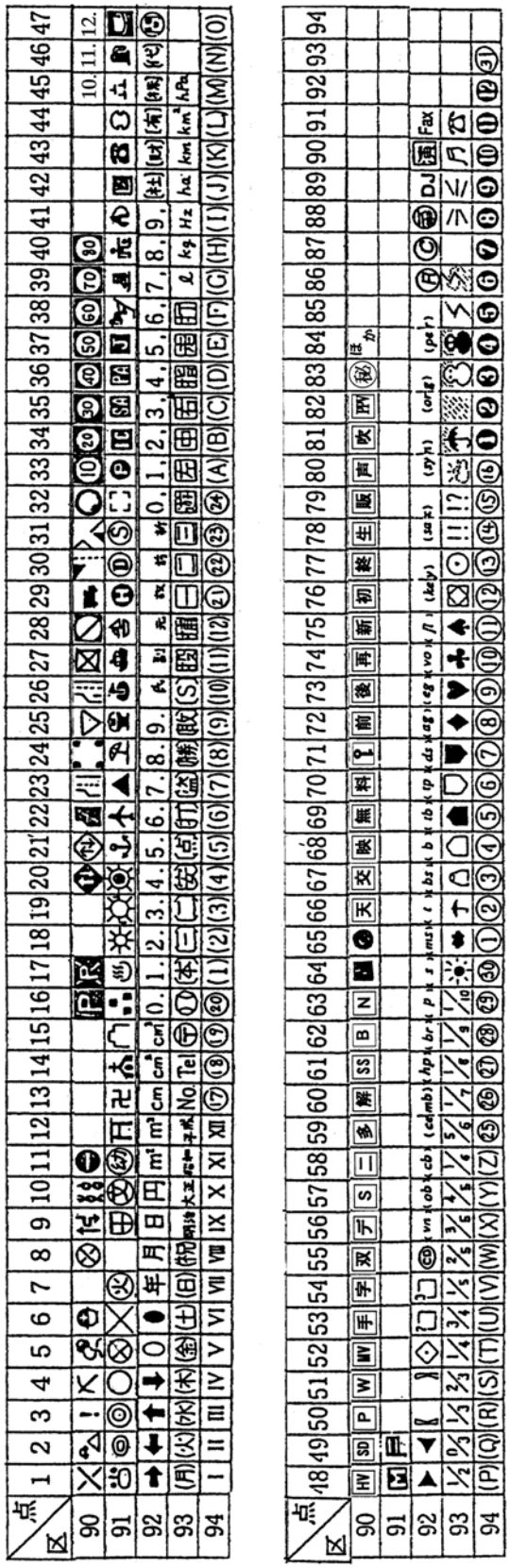


Table E-4 Code position of additional characters for broadcasting

Cell Row	1	2	3	4	5	6	7	8	9	10	11	12
90	F803	F804	F805	F806	F807	F808		F809	F80A	F80B	F80C	
91	F848	F849	F84A	F84B	F84C	F84D	F84E		F84F	F850	F851	F852
92	F877	F878	F879	F87A	F87B	F87C	F87D	F87E	F87F	F880	33A1	33A5
93	322A	322B	322C	322D	322E	322F	3230	3237	337E	337D	337C	337B
94	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	216A	216B

Cell Row	13	14	15	16	17	18	19	20	21	22	23	24
90				F80D	F80E			F80F	F810	F811	F812	F813
91	F853	F854	F855	F856	2668	F857	F858	F859	F85A	F85B	F85C	F85D
92	339D	33A0	33A4	F881	2488	2489	248A	248B	248C	248D	248E	248F
93	F8B9	F8BA	3036	F8BB	F8BC	F8BD	F8BE	F8BF	F8C0	F8C1	F8C2	F8C3
94	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479	247A	247B

Cell Row	25	26	27	28	29	30	31	32	33	34	35	36
90	F814	F815	F816	F817	F818	F819	F81A	F81B	F81C	F81D	F81E	F81F
91	F85E	F85F	F860	F861	F862	F863	F864	F865	F866	F867	F868	F869
92	2490	F882	F883	F884	F885	F886	F887	F888	F889	F88A	F88B	F88C
93	F8C4	F8C5	F8C6	F8C7	F8C8	F8C9	F8CA	F8CB	F8CC	F8CD	F8CE	F8CF
94	247C	247D	247E	247F	3251	3252	3253	3254	F8E6	F8E7	F8E8	F8E9

Cell Row	37	38	39	40	41	42	43	44	45	46	47	48
90	F820	F821	F822	F823					2491	2492	2493	F824
91	F86A	F86B	F86C	F86D	F86E	F86F	F870	F871	F872	F873	F874	F875
92	F88D	F88E	F88F	F890	F891	3233	3236	3232	3231	3239	F892	25B6
93	F8D0	F8D1	2113	338F	3390	33CA	339E	33A2	3371			00BD
94	F8EA	F8EB	F8EC	F8ED	F8EE	F8EF	F8F0	F8F1	F8F2	F8F3	F8F4	F8F5

Cell Row	49	50	51	52	53	54	55	56	57	58	59	60
90	F825	F826	F827	F828	F829	F82A	F82B	F82C	F82D	F82E	F82F	F830
91	F876											
92	25C0	3016	3017	F893	F894	F895	F896	F897	F898	F899	F89A	F89B
93	F8D2	2153	2154	00BC	00BE	2155	2156	2157	2158	2159	215A	F8D3
94	F8F6	F8F7	F8F8	F8F9	F8FA	F8FB	F8FC	F8FD	F8FE	F8FF	3255	3256

Cell Row	61	62	63	64	65	66	67	68	69	70	71	72
90	F831	F832	F833	F834	F835	F836	F837	F838	F839	F83A	F83B	F83C
91												
92	F89C	F89D	F89E	F89F	F8A0	F8A1	F8A2	F8A3	F8A4	F8A5	F8A6	F8A7
93	215B	F8D4	F8D5	2600	2601	2602	F8D6	F8D7	F8D8	F8D9	F8DA	2666
94	3257	3258	3259	325A	2460	2461	2462	2463	2464	2465	2466	2467

Cell	73	74	75	76	77	78	79	80	81	82	83	84

Row												
90	F83D	F83E	F83F	F840	F841	F842	F843	F844	F845	F846	3299	F847
91												
92	F8A8	F8A9	F8AA	F8AB	F8AC	F8AD	F8AE	F8AF	F8B0	F8B1	F8B2	F8B3
93	2665	2663	2660	F8DB	F8DC	203C	2049	F8DD	F8DE	F8DF	F8E0	F8E1
94	2468	2469	246A	246B	246C	246D	246E	246F	2776	2777	2778	2779

Cell Row	85	86	87	88	89	90	91	92	93	94		
90												
91												
92	F8B4	00AE	00A9	F8B5	F8B6	F8B7	F8B8					
93	F8E2	F8E3		F8E4	F8E5	266C	260E					
94	277A	277B	277C	277D	277E	277F	24EB	24EC	325B			

Annex F Void

Annex G Minimum platform capabilities

G.1 Graphics

This section conforms to GEM1.0 “G.1 Graphics”. The resolution of each plane conforms to “13.1 Corresponding resolution”.

G.1.1 Resolution

In this standard, the resolution “720x576” shall conform to the resolution described in ARIBSTD-B21.

G.1.2 CLUT

As described in Chapter 15 Table 15-1 “Minimum CLUT”.

G.2 Audio

This section conforms to GEM1.0 “G.2 Audio”.

G.3 Video

This section conforms to GEM1.0 “G.3 Video”.

G.4 Resident fonts and text rendering

This section conforms to GEM1.0 “G.4 Resident fonts and text rendering”.

G.5 Input events

This section conforms to GEM1.0 “G.5 Input events”. Note that VK_TELETEXT shall not be used. For mapping of actual buttons to remote controller and so on, it shall be specified in operational guideline.

G.6 Memory

This section conforms to GEM1.0 “G.6 Memory”.

G.7 Other resources

This section conforms to GEM1.0 “G.7 Other resources”.

Annex H Extensions

This Annex specifies the rules for API to be extended in the future and important notices for the case of future extension.

It is not allowed to add any constructors, methods or fields of which attribute are “public” or “protected” for the classes and interfaces in jp.or.arib.tv package.

The standards other than those mentioned above shall conform to GEM1.0 “Annex H: Extensions”.

Annex I ARIB-J fundamental classes

This Annex defines ARIB-J basic class and conforms to GEM1.0 “Annex I: DVB-J fundamental classes”.

Annex J ARIB-J event API

This Annex specifies API relevant to event operation for general-purpose to be mainly used in user interfaces in ARIB-J.
This Annex conforms to GEM1.0 “Annex J: DVB-J event API”.

Annex K ARIB-J persistent storage API

This Annex specifies API relevant to the operation of persistent storage area to be implemented such as non-volatile memory in ARIB-J.
This Annex conforms to GEM1.0 “Annex K: DVB-J persistent storage API”.

Annex L User setting and preferences API

This Annex specifies API relevant to the operation for end-user setting information to be stored in the receiver. This Annex conforms to GEM1.0 “Annex L: User settings and preferences API”.

Annex M SI Access API

Additional definitions for jp.or.arib.tv.net package and jp.or.arib.tv.si package are provided.

M.1 Package jp.or.arib.tv.net
See ARIB – specific contents.

General information on class	
ARIBLocator	ARIB Locator encapsulates ARIB URL into the object.
ARIBNetworkBoundLocator	ARIBLocator connects to the network

M.1.1 Description of package jp.or.arib.tv.net

See ARIB-specific contents.

M.1.2 ARIBLocator
jp.or.arib.tv.net
Class ARIBLocator

java.lang.Object
  |
  +--org.davic.net.Locator
  |
  +--**jp.or.arib.tv.net.ARIBLocator**

Direct and known subclass:
ARIBNetworkBoundLocator

public class **ARIBLocator**
extends org.davic.net.Locator

ARIB Locator encapsulates ARIB URL into the object.

General information on the constructor
ARIBLocator(java.lang.String url) Generation of the ARIB Locator
ARIBLocator(java.lang.String scheme, int onid, int tsid) Generation of the ARIB Locator based on the following format.
ARIBLocator(java.lang.String scheme, int onid, int tsid, int serviceid) Generation of the ARIB Locator based on the following format.
ARIBLocator(java.lang.String scheme, int onid, int tsid, int serviceid, int contentid) Generation of the ARIB Locator based on the following format.
ARIBLocator(java.lang.String scheme, int onid, int tsid, int serviceid, int contentid, int eventid)

Generation of the ARIB Locator based on the following format.
ARIBLocator(java.lang.String scheme, int onid, int tsid, int serviceid, int contentid, int eventid, int componenttag) Generation of the ARIB Locator based on any of the following formats.
ARIBLocator(java.lang.String scheme, int onid, int tsid, int serviceid, int contentid, int eventid, int[] componenttags) Generation of the ARIB Locator based on any of the following formats.
ARIBLocator(java.lang.String scheme, int onid, int tsid, int serviceid, int contentid, int eventid, int[] componenttags, java.lang.String filePath) Generation of the ARIB Locator based on any of the following formats.
ARIBLocator(java.lang.String scheme, int onid, int tsid, int serviceid, int contentid, int eventid, int componenttag, int channelid) Generation of the ARIB Locator based on any of the following formats.
ARIBLocator(java.lang.String scheme, int onid, int tsid, int serviceid, int contentid, int eventid, int componenttag, java.lang.String modulename) Generation of the ARIB Locator based on any of the following formats.
ARIBLocator(java.lang.String scheme, int onid, int tsid, int serviceid, int contentid, int eventid, int componenttag, java.lang.String modulename, java.lang.String resourcename) Generation of the ARIB Locator based on any of the following formats.

General information on the method	
int	**getChannelId** () To acquire channel_id.
int[]	**getComponentTags**() To acquire the array of component_tag.
int	**getContentId**() To acquire content_id.
int	**getEventId**() To acquire event_id.
java.lang.String	**getFilePath**() To acquire the path part of the locator's file name.
java.lang.String	**getModuleName**() To acquire moduleName.
int	**getOriginalNetworkId**() To acquire original_network_id.
java.lang.String	**getResourceName**() To acquire resourceName.
java.lang.String	**getScheme**() To acquire the scheme.
int	**getServiceId**() To acquire service_id.
int	**getTransportStreamId**() To acquire transport_stream_id.

java.net.URL	**getURL**() To acquire ARIB URL capsulated into the ARIBLocator object.

Method inherited from the class org.davic.net.Locator
hasMultipleTransformations, toExternalForm, toString

Method inherited from the class, java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Details on the constructor

ARIBLocator
public **ARIBLocator**(java.lang.String url)
throws org.davic.net.InvalidLocatorException

Generation of the ARIB Locator.

Parameter:

url – URL letter string

Exception:

org.davic.net.InvalidLocatorException – In case the proper locator cannot be generated with the assigned parameter (e.g. a numerical value outside the effective range).

ARIBLocator
public **ARIBLocator**(java.lang.String scheme,
int onid,
int tsid)
throws org.davic.net.InvalidLocatorException

Generation of the ARIB Locator based on the following format. “scheme: //onid.tsid”

Parameter:

scheme – locator scheme
onid – original network ID
tsid – transport stream ID

Exception:

org.davic.net.InvalidLocatorException - In case the proper locator cannot be generated with the assigned parameter (e.g. a numerical value outside the effective range).

ARIBLocator
public **ARIBLocator** (java.lang.String scheme,
int onid,

int tsid,
int serviceid)
throws org.davic.net.InvalidLocatorException

Generation of the ARIB Locator based on the following format.
"scheme: //onid.tsid.serviceid"

Parameter:

scheme – locator scheme
onid – original network ID
tsid – transport stream ID
serviceid – service ID

Exception:

org.davic.net.InvalidLocatorException - In case the proper locator cannot be generated with the assigned parameter (e.g. a numerical value outside the effective range).

ARIBLocator
public **ARIBLocator**(java.lang.String scheme,
int onid,
int tsid,
int serviceid,
int contentid)
throws org.davic.net.InvalidLocatorException

Generation of the ARIB Locator based on the following format.
"scheme: //onid.tsid.serviceid;contentId"

Parameter:

scheme - locator scheme
onid - original network ID
tsid - transport stream ID (When the transport stream ID is not included, the value is "1".)
serviceid - service ID
contentid – contents ID

Exception:

org.davic.net.InvalidLocatorException - In case the proper locator cannot be generated with the assigned parameter (e.g. a numerical value outside the effective range)

ARIBLocator
public **ARIBLocator**(java.lang.String scheme,
int onid,
int tsid,
int serviceid,
int contentid,

int eventid)
throws org.davic.net.InvalidLocatorException

Generation of the ARIB Locator based on the following format. “scheme: //onid.tsid.serviceid;contentId.eventid”

Parameter:

scheme - locator scheme
onid - original network ID
tsid - transport stream ID (When the transport stream ID is not included, the value is “1”.)
serviceid - service ID
contentid - contents ID (When the contents ID is not included, the value is “1”.)
eventid – event ID

Exception:

org.davic.net.InvalidLocatorException - In case the proper locator cannot be generated with the assigned parameter (e.g. a numerical value outside the effective range)

ARIBLocator
public **ARIBLocator**(java.lang.String scheme,
int onid,
int tsid,
int serviceid,
int contentid,
int eventid,
int componenttag)
throws org.davic.net.InvalidLocatorException

Generation of the ARIB Locator based on any of the following formats.
“scheme: //onid.tsid.serviceid;contentid.eventid/componenttag”
“scheme: //onid.tsid.serviceid;contentId/componenttag”
“scheme: //onid.tsid.serviceid.eventId/componenttag”

Parameter:

scheme - locator scheme
onid - original network ID
tsid - transport stream ID
serviceid - service ID
contentid - contents ID (When the contents ID is not included, the value is “1”.)
eventid - event ID (When the event ID is not included, the value is “1”.)
componenttag – component tag

Exception:

org.davic.net.InvalidLocatorException - In case the proper locator cannot be generated with the assigned parameter (e.g. a numerical value outside the effective range).

ARIBLocator
public **ARIBLocator**(java.lang.String scheme,
int onid,
int tsid,
int serviceid,
int contentid,
int eventid,
int[] componenttags)
throws org.davic.net.InvalidLocatorException

Generation of the ARIB Locator based on any of the following formats.
"scheme: //onid.tsid.serviceid;contentid.eventid/componenttag{&componenttag}"
"scheme: //onid.tsid.serviceid;contentid/componenttag{&componenttag}"
"scheme: //onid.tsid.serviceid.eventid/componenttag{&componenttag}"

Parameter:

scheme - locator scheme
onid - original network ID
tsid - transport stream ID
serviceid - service ID
contentid - contents ID (When the contents ID is not included, the value is "-1".)
eventid - event ID (When the event ID is not included, the value is "-1".)
componenttags – array of component tag

Exception:

org.davic.net.InvalidLocatorException - In case the proper locator cannot be generated with the assigned parameter (e.g. a numerical value outside the effective range)

ARIBLocator
public **ARIBLocator**(java.lang.String scheme,
int onid,
int tsid,
int serviceid,
int contentid,
int eventid,
int[] componenttags,
java.lang.String filePath)
throws org.davic.net.InvalidLocatorException

Generation of the ARIB Locator based on any of the following formats.
"scheme:
//onid.tsid.serviceid;contentid.eventid/componenttag{&componenttag}/filepath"
"scheme: //onid.tsid.serviceid;contentid/componenttag{&componenttag}/filepath"

"scheme: //onid.tsid.serviceid.eventid/componenttag{&componenttag}/filepath"

Parameter:

scheme - locator scheme
onid - original network ID
tsid - transport stream ID
serviceid - service ID
eventid - event ID (When the event ID is not included, the value is "-1".)
componenttags - array of component tag (When the component tag is not included, the value is "null".)
filePath – file path string that starts with a stroke

Exception:

org.davic.net.InvalidLocatorException - In case the proper locator cannot be generated with the assigned parameter (e.g. a numerical value outside the effective range)

ARIBLocator
public **ARIBLocator**(java.lang.String scheme,
int onid,
int tsid,
int serviceid,
int contentid,
int eventid,
int componenttag,
int channelid)
throws org.davic.net.InvalidLocatorException

Generation of the ARIB Locator based on any of the following formats.
"scheme: //onid.tsid.serviceid;contentid.eventid/componenttag;channelid"
"scheme: //onid.tsid.serviceid;contentId/componenttag;channelid"
"scheme: //onid.tsid.serviceid.eventId/componenttag;channelid"

Parameter:

scheme - locator scheme
onid - original network ID
tsid - transport stream ID
serviceid - service ID
contentid - contents ID (When the contents ID is not included, the value is "-1".)
eventid - event ID (When the event ID is not included, the value is "-1".)
componenttag – component tag
channelid – dual monaural audio channel ID (1-the first channel, 2-the second channel, 3-synchronized replay by both channels)

Exception:

org.davic.net.InvalidLocatorException - In case the proper locator cannot be generated

with the assigned parameter (e.g. a numerical value outside the effective range) .

ARIBLocator
public **ARIBLocator**(java.lang.String scheme,
int onid,
int tsid,
int serviceid,
int contentid,
int eventid,
int componenttag,
java.lang.String modulename)
throws org.davic.net.InvalidLocatorException

Generation of the ARIB Locator based on any of the following formats.
"scheme: //onid.tsid.serviceid;contentid.eventid/componenttag/modulename"
"scheme: //onid.tsid.serviceid;contentid/componenttag/modulename"
"scheme: //onid.tsid.serviceid.eventid/componenttag/modulename"

Parameter:

scheme - locator scheme
onid - original network ID
tsid - transport stream ID
serviceid - service ID
eventid - event ID (When the event ID is not included, the value is "-1".)
componenttag – component tag
modulename – module name or module ID

Exception:

org.davic.net.InvalidLocatorException - In case the proper locator cannot be generated with the assigned parameter (e.g. a numerical value outside the effective range)

ARIBLocator
public **ARIBLocator**(java.lang.String scheme,
int onid,
int tsid,
int serviceid,
int contentid,
int eventid,
int componenttag,
java.lang.String modulename,
java.lang.String resourcename)
throws org.davic.net.InvalidLocatorException

Generation of the ARIB Locator based on any of the following formats. "scheme:

//onid.tsid.serviceid;contentid.eventid/componenttag/modulename/resourcename"
"scheme: //onid.tsid.serviceid;contentid/componenttag/modulename/resourcename"
"scheme: //onid.tsid.serviceid.eventid/componenttag/modulename/resourcename"

Parameter:

scheme - locator scheme
onid - original network ID
tsid - transport stream ID
serviceid - service ID
eventid - event ID (When the event ID is not included, the value is "-1".)
componenttag – component tag
modulename - module name or module ID

Exception:

org.davic.net.InvalidLocatorException - In case the proper locator cannot be generated with the assigned parameter (e.g. a numerical value outside the effective range)

Details of the method

getChannelId
public int **getChannelId**()

To acquire channel_id.

Return value:

channel_id. If it is absent, the value is "-1".

getComponentTags
public int[] **getComponentTags**()

To acquire the array of the component_tag.

Return value:

The array that includes the component_tag. If this locator does not identify the component_tag, an array with the length "0" is returned.

getContentId
public int **getContentId**()

To acquire content_id.

Return value:

content_id. If it is absent, the value is "-1".

getEventId
public int **getEventId**()

To acquire event_id.

Return value:

event_id. If it is absent, the value is "-1".

getFilePath
public java.lang.String **getFilePath**()

To acquire the path part of the locator's file name.

Return value:

File path string that starts with a stroke. If the locator does not include the path string, this method returns "null".

getModuleName
public java.lang.String **getModuleName**()

To acquire moduleName.

Return value:

moduleName. If it is absent, the value is "null".

getOriginalNetworkId
public int **getOriginalNetworkId**()

To acquire original_network_id.

Return value:

original_network_id. If it is absent, the value is "-1".

getResourceName
public java.lang.String **getResourceName**()

To acquire resourceName.

Return value:

resourceName. If it is absent, the value is "null".

getScheme
public java.lang.String **getScheme**()

To acquire the scheme.

Return value:

Scheme. If it is absent, the value is "null".

getServiceId
public int **getServiceId**()

To acquire service_id.

Return value:

service_id. If it is absent, the value is "-1".

getTransportStreamId
public int **getTransportStreamId**()

To acquire transport_stream_id.

Return value:

transport_stream_id. If it is absent, the value is “-1”.

getURL

public java.net.URL **getURL**()

To acquire ARIB URL capsulated into the ARIBLocator object.

Return value:

URL

M.1.3 ARIBNetworkBoundLocator

jp.or.arib.tv.net

Class ARIBNetworkBoundLocator

java.lang.Object
 |
 +--org.davic.net.Locator
 |
 +--jp.or.arib.tv.net.ARIBLocator
 |
 +--**jp.or.arib.tv.net.ARIBNetworkBoundLocator**

Mounted interface for all:

org.davic.net.TransportDependentLocator

public class **ARIBNetworkBoundLocator**
extends ARIBLocator
implements org.davic.net.TransportDependentLocator

ARIBLocator is connected with the network. This type of object uniquely identifies a certain entity including the distribution system that transmits the entity.

For example, if a certain service is transmitted via two types of networks, i.e. satellite broadcasting and terrestrial broadcasting, the service may be identified as a common service in the ARIBLocator. However, each transmitted service has a different ARIBNetworkBoundLocator from each other.

General information on the constructor
ARIBNetworkBoundLocator(ARIBLocator unboundLocator, int networkId) Generation of the network bound locator.

General information on the method	
int	**getNetworkId**() To acquire network_id.

Method inherited from the class jp.or.arib.tv.net.ARIBLocator
getComponentTags, getContentId, getEventId, getFilePath, getModuleName, getOriginalNetworkId, getResourceName, getScheme, getServiceId, getTransportStreamId, getURL

Method inherited from the class org.davic.net.Locator
hasMultipleTransformations, toExternalForm, toString

Method inherited from the class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Details of the constructor

ARIBNetworkBoundLocator

public **ARIBNetworkBoundLocator**(ARIBLocator unboundLocator,
int networkId)
throws org.davic.net.InvalidLocatorException

Generation of the network bound locator.

Parameter:

unboundLocator – network unbound ARIB locator

networkId – network ID

Exception:

org.davic.net.InvalidLocatorException - In case the proper locator cannot be generated with the assigned parameter (e.g. a numerical value outside the effective range).

Details of the method

getNetworkId

public int **getNetworkId**()

To acquire network_id.

Return value:

network_id

M.2 Package jp.or.arib.tv.si
Access to ARIB Service information.

General information on the interface	
DescriptorTag	This interface defines the constant that corresponds to the most common tag value.
PMTElementaryStream	This interface indicates the elementary stream of the service (channel).
PMTService	This interface indicates the specific service that is transmitted by transport stream.
PMTStreamType	This interface defines the constant that corresponds to the various stream format.
SIBouquet	This interface indicates the sub-table of Bouquet Association Table (BAT) that described the specific bouquet (with the SITransportStreamBAT interface)
SIBroadcaster	This interface indicates the specific broadcaster in the service.
SIEvent	This interface indicates the specific program in the service.
SIInformation	This interface is a collection of common functions of SIBouquet, SIBroadcaster, SINetwork, SITransportStream, SIService, PMTService, SIEvent, SITime and PMTElementaryStream.
SIIterator	The object mounted with the SIIterator interface enables access to the contents via the collection of the SI object collection.
SIMonitoringListener	This is an interface to be mounted by application class for receipt of the monitoring SI object change.
SIMonitoringType	This interface defines the constant that corresponds to each item of the SI information in a SIMonitoringEvent.
SINetwork	This interface indicates the sub-table of the Network Information Table (NIT) that describes the specific network (with the SITransportStreamNIT).
SIRetrievalListener	This interface must be mounted by the application in order to receive the completed event of the SI request.
SIRunningStatus	This interface defines the constant that corresponds to the progressing value in the service and event.
SIService	This interface indicates a specific service that is transmitted by one transport stream.
SIServiceType	Constant definition that indicates the service type classification.
SITime	This indicates time date table (TDT) and (optional) time date offset table.
SITransportStream	This is the base interface to show the information relevant to the transport stream.
SITransportStreamBAT	This indicates the information relevant to the transport stream acquired from BAT.
SITransportStreamNIT	This indicates the information relevant to the transport stream

	acquired from NTN.

General information on class	
Descriptor	This class indicates the descriptor inside the sub-table.
SIDatabase	This class indicates the root of the SI hierarchy structure.
SIExEventInformation	This interface indicates the description items of the details regarding specific program and the item names.
SILackOfResourcesEvent	This event is informed when the necessary resources for data acquisition are not available at the time of the SI acquisition request.
SIMonitoringEvent	This class object is transmitted to the listener object so as to inform the application of the change of monitored information.
SINotInCacheEvent	This event is informed as a reply when SI acquisition is requested in FROM_CACHE_ONLY mode and the requested data are absent in the cache.
SIObjectNotInTableEvent	This event is informed when an SI table that has the position information of the requested object is acquired for the purpose of the SI acquisition request but the object is absent.
SIRequest	This instance object of this class indicates the acquisition request from the application.
SIRequestCancelledEvent	This event is transmitted as a reply to the SI acquisition request when the request was canceled by calling SIRequest cancelRequest method.
SIRetrievalEvent	This is the basic class for the event regarding SI acquisition request completion.
SISuccessfulRetrieveEvent	This event is transmitted as a reply to the SI acquisition request when the SI acquisition is completed properly.
SITableNotFoundEvent	This event is transmitted as a reply to the SI acquisition request when the SI table that has the requested information cannot be acquired.
SITableUpdatedEvent	This event is transmitted as a reply to the SI acquisition request when the table that transmits the information about the said object has been updated and the descriptor information consistent with the old object cannot be acquired.
SIUtil	This class includes SI relevant utility function.

General information on exceptions	
SIException	This class is the root of the SI hierarchy structure.
SIIllegalArgumentException	This exemption occurs when more than one inappropriate parameter is given to the method.
SIInvalidPeriodException	The exception occurs when the designated time period is inappropriate.

M.2.1 Description of package jp.or.arib.tv.si

Access to ARIB service information and low-level SI API equivalent to org.dvb.si package in MHP

This package is a revised or extended one in accordance with the ARIB service information specified by ARIB STD-B10. The overall structure of API and the usage are almost the same as org.dvb.si.

M.2.2 Descriptor tag
jp.or.arib.tv.si
Interface DescriptorTag

public interface **DescriptorTag**
This interface defines the constant that corresponds to the most common descriptor tag value.

Reference:
Descriptor

General information on the field	
static short	**AUDIO_COMPONENT** The constant indicates the descriptor tag value of the audio component descriptor specified in ARIB STD-B10.
static short	**BASIC_LOCAL_EVENT** The constant indicates the descriptor tag value of the basic local event descriptor specified in ARIB STD-B10.
static short	**BOARD_INFORMATION** The constant indicates the descriptor tag value of the board information descriptor specified in ARIB STD-B10.
static short	**BOUQUET_NAME** The constant indicates the descriptor tag value of the bouquet name descriptor specified in ARIB STD-B10.
static short	**BROADCASTER_NAME** The constant indicates the descriptor tag value of the broadcaster name descriptor specified in ARIB STD-B10.
static short	**CA_CONTRACT_INFO** The constant indicates the descriptor tag value of the CA contractor information descriptor specified in ARIB STD-B10.
static short	**CA_EMM_TS** The constant indicates the descriptor tag value of the CA_EMM_TS descriptor specified in ARIB STD-B10.
static short	**CA_IDENTIFIER** The constant indicates the descriptor tag value of the CA identification descriptor specified

	in ARIB STD-B10.
static short	**CA_SERVICE** The constant indicates the descriptor tag value of the CA service descriptor specified in ARIB STD-B10.
static short	**CABLE_DELIVERY_SYSTEM** The constant indicates the descriptor tag value of the cable delivery system descriptor specified in ARIB STD-B10.
static short	**CAROUSEL_COMPATIBLE_COMPOSITE** The constant indicates the descriptor tag value of the carousel compatible composite descriptor specified in ARIB STD-B10.
static short	**COMPONENT** The constant indicates the descriptor tag value of the component descriptor specified in ARIB STD-B10.
static short	**COMPONENT_GROUP** The constant indicates the descriptor tag value of the component group descriptor specified in ARIB STD-B10.
static short	**CONNECTED_TRANSMISSION** The constant indicates the descriptor tag value of the connected transmission descriptor specified in ARIB STD-B10.
static short	**CONTENT** The constant indicates the descriptor tag value of the content descriptor specified in ARIB STD-B10.
static short	**CONTENT_AVAILABILITY** The constant indicates the descriptor tag value of the contents availability descriptor specified in ARIB STD-B10.
static short	**COUNTRY_AVAILABILITY** The constant indicates the descriptor tag value of the country receiving availability descriptor specified in ARIB STD-B10.
static short	**DATA_COMPONENT** The constant indicates the descriptor tag value of the data component descriptor specified in ARIB STD-B10.
static short	**DATA_CONTENTS** The constant indicates the descriptor tag value of the data contents descriptor specified in ARIB STD-B10.
static short	**DIGITAL_COPY_CONTROL** The constant indicates the descriptor tag value of the digital copy control descriptor specified in ARIB STD-B10.
static short	**DOWNLOAD_CONTENT** The constant indicates the descriptor tag value of the download contents descriptor specified in ARIB STD-B10.
static short	**EMERGENCY_INFORMATION** The constant indicates the descriptor tag value of the emergency information descriptor specified in ARIB STD-B10.
static short	**EVENT_GROUP** The constant indicates the descriptor tag value of the event group descriptor specified in ARIB STD-B10.

static short	**EXTENDED_BROADCASTER** The constant indicates the descriptor tag value of the extended broadcaster descriptor specified in ARIB STD-B10.
static short	**EXTENDED_EVENT** The constant indicates the descriptor tag value of the extended event descriptor specified in ARIB STD-B10.
static short	**HIERARCHICAL_TRANSMISSION** The constant indicates the descriptor tag value of the hierarchical transmission descriptor specified in ARIB STD-B10.
static short	**HYPER_LINK** The constant indicates the descriptor tag value of the hyper link descriptor specified in ARIB STD-B10.
static short	**LDT_LINKAGE** The constant indicates the descriptor tag value of the LDT linkage descriptor specified in ARIB STD-B10.
static short	**LINKAGE** The constant indicates the descriptor tag value of the linkage descriptor specified in ARIB STD-B10.
static short	**LOCAL_TIME_OFFSET** The constant indicates the descriptor tag value of the local time offset descriptor specified in ARIB STD-B10.
static short	**LOGO_TRANSMISSION** The constant indicates the descriptor tag value of the logo transmission descriptor specified in ARIB STD-B10.
static short	**MOSAIC** The constant indicates the descriptor tag value of the mosaic descriptor specified in ARIB STD-B10.
static short	**NETWORK_IDENTIFICATION** The constant indicates the descriptor tag value of the network identification descriptor specified in ARIB STD-B10.
static short	**NETWORK_NAME** The constant indicates the descriptor tag value of the network name descriptor specified in ARIB STD-B10.
static short	**NODE_RELATION** The constant indicates the descriptor tag value of the node relation descriptor specified in ARIB STD-B10.
static short	**NVOD_REFERENCE** The constant indicates the descriptor tag value of the NVOD reference service descriptor specified in ARIB STD-B10.
static short	**PARENTAL_RATING** The constant indicates the descriptor tag value of the parental rating descriptor specified in ARIB STD-B10.
static short	**PARTIAL_RECEPTION** The constant indicates the descriptor tag value of the partial reception descriptor specified in ARIB STD-B10.
static short	**PARTIAL_TRANSPORT_STREAM**

	The constant indicates the descriptor tag value of the partial transport stream descriptor specified in ARIB STD-B10.
static short	**PARTIALTS_TIME** The constant indicates the descriptor tag value of the partial transport stream time descriptor specified in ARIB STD-B10.
static short	**REFERENCE** The constant indicates the descriptor tag value of the reference descriptor specified in ARIB STD-B10.
static short	**SATELLITE_DELIVERY_SYSTEM** The constant indicates the descriptor tag value of the satellite delivery system descriptor specified in ARIB STD-B10.
static short	**SERIES** The constant indicates the descriptor tag value of the series descriptor specified in ARIB STD-B10.
static short	**SERVICE** The constant indicates the descriptor tag value of the service descriptor specified in ARIB STD-B10.
static short	**SERVICE_LIST** The constant indicates the descriptor tag value of the service list descriptor specified in ARIB STD-B10.
static short	**SHORT_EVENT** The constant indicates the descriptor tag value of the short form event descriptor specified in ARIB STD-B10.
static short	**SHORT_NODE_INFORMATION** The constant indicates the descriptor tag value of the short form node information descriptor specified in ARIB STD-B10.
static short	**SI_PARAMETER** The constant indicates the descriptor tag value of the SI transmission parameter descriptor specified in ARIB STD-B10.
static short	**SI_PRIME_TS** The constant indicates the descriptor tag value of the SI prime TS descriptor specified in ARIB STD-B10.
static short	**STC_REFERENCE** The constant indicates the descriptor tag value of the STC reference descriptor specified in ARIB STD-B10.
static short	**STREAM_IDENTIFIER** The constant indicates the descriptor tag value of the stream identification descriptor specified in ARIB STD-B10.
static short	**STUFFING** The constant indicates the descriptor tag value of the stuffing descriptor specified in ARIB STD-B10.
static short	**SYSTEM_MANAGEMENT** The constant indicates the descriptor tag value of the system management descriptor specified in ARIB STD-B10.
static short	**TARGET_AREA** The constant indicates the descriptor tag value of the target area descriptor specified in

	ARIB STD-B10.
static short	**TERRESTRIAL_DELIVERY_SYSTEM** The constant indicates the descriptor tag value of the terrestrial delivery system descriptor specified in ARIB STD-B10.
static short	**TIME_SHIFTED_EVENT** The constant indicates the descriptor tag value of the time shifted event descriptor specified in ARIB STD-B10.
static short	**TIME_SHIFTED_SERVICE** The constant indicates the descriptor tag value of the time shifted service descriptor specified in ARIB STD-B10.
static short	**TS_INFORMATION** The constant indicates the descriptor tag value of the TS information descriptor specified in ARIB STD-B10.
static short	**VIDEO_DECODE_CONTROL** The constant indicates the descriptor tag value of the video decode control descriptor specified in ARIB STD-B10.

Details of the field

AUDIO_COMPONENT

public static final short **AUDIO_COMPONENT**

The constant indicates the descriptor tag value of the audio component descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

BASIC_LOCAL_EVENT

public static final short **BASIC_LOCAL_EVENT**

The constant indicates the descriptor tag value of the basic local event descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

BOARD_INFORMATION

public static final short **BOARD_INFORMATION**

The constant indicates the descriptor tag value of the board information descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

BOUQUET_NAME

public static final short **BOUQUET_NAME**

The constant indicates the descriptor tag value of the bouquet name descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

BROADCASTER_NAME

public static final short **BROADCASTER_NAME**

The constant indicates the descriptor tag value of the broadcaster name descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

CA_CONTRACT_INFO

public static final short **CA_CONTRACT_INFO**

The constant indicates the descriptor tag value of the CA contractor information descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

CA_EMM_TS

public static final short **CA_EMM_TS**

The constant indicates the descriptor tag value of the CA_EMM_TS descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

CA_IDENTIFIER

public static final short **CA_IDENTIFIER**

The constant indicates the descriptor tag value of the CA identification descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

CA_SERVICE

public static final short **CA_SERVICE**

The constant indicates the descriptor tag value of the CA service descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

CABLE_DELIVERY_SYSTEM

public static final short **CABLE_DELIVERY_SYSTEM**

The constant indicates the descriptor tag value of the cable delivery system descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

CAROUSEL_COMPATIBLE_COMPOSITE

public static final short **CAROUSEL_COMPATIBLE_COMPOSITE**

The constant indicates the descriptor tag value of the carousel compatible composite descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

COMPONENT

public static final short **COMPONENT**

The constant indicates the descriptor tag value of the component descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

COMPONENT_GROUP

public static final short **COMPONENT_GROUP**

The constant indicates the descriptor tag value of the component group descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

CONNECTED_TRANSMISSION

public static final short **CONNECTED_TRANSMISSION**

The constant indicates the descriptor tag value of the connected transmission descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

CONTENT

public static final short **CONTENT**

The constant indicates the descriptor tag value of the content descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

CONTENT_AVAILABILITY
public static final short **CONTENT_AVAILABILITY**

The constant indicates the descriptor tag value of the contents availability descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

COUNTRY_AVAILABILITY
public static final short **COUNTRY_AVAILABILITY**

The constant indicates the descriptor tag value of the country receiving availability descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

DATA_COMPONENT
public static final short **DATA_COMPONENT**

The constant indicates the descriptor tag value of the data contents descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

DATA_CONTENTS
public static final short **DATA_CONTENTS**

The constant indicates the descriptor tag value of the data contents descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

DIGITAL_COPY_CONTROL
public static final short **DIGITAL_COPY_CONTROL**

The constant indicates the descriptor tag value of the digital copy control descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

DOWNLOAD_CONTENT
public static final short **DOWNLOAD_CONTENT**

The constant indicates the descriptor tag value of the download contents descriptor

specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

EMERGENCY_INFORMATION

public static final short **EMERGENCY_INFORMATION**

The constant indicates the descriptor tag value of the emergency information descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

EVENT_GROUP

public static final short **EVENT_GROUP**

The constant indicates the descriptor tag value of the event group descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

EXTENDED_BROADCASTER

public static final short **EXTENDED_BROADCASTER**

The constant indicates the descriptor tag value of the extended broadcaster descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

EXTENDED_EVENT

public static final short **EXTENDED_EVENT**

The constant indicates the descriptor tag value of the extended event descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

HIERARCHICAL_TRANSMISSION

public static final short **HIERARCHICAL_TRANSMISSION**

The constant indicates the descriptor tag value of the hierarchical transmission descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

HYPER_LINK
public static final short **HYPER_LINK**

The constant indicates the descriptor tag value of the hyper link descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

LDT_LINKAGE
public static final short **LDT_LINKAGE**

The constant indicates the descriptor tag value of the LDT linkage descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

LINKAGE
public static final short **LINKAGE**

The constant indicates the descriptor tag value of the linkage descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

LOCAL_TIME_OFFSET
public static final short **LOCAL_TIME_OFFSET**

The constant indicates the descriptor tag value of the local time offset descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

LOGO_TRANSMISSION
public static final short **LOGO_TRANSMISSION**

The constant indicates the descriptor tag value of the logo transmission descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

MOSAIC
public static final short **MOSAIC**

The constant indicates the descriptor tag value of the mosaic descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

NETWORK_IDENTIFICATION
public static final short **NETWORK_IDENTIFICATION**

The constant indicates the descriptor tag value of the network identification descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

NETWORK_NAME
public static final short **NETWORK_NAME**

The constant indicates the network name descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

NODE_RELATION
public static final short **NODE_RELATION**

The constant indicates the descriptor tag value of the node relation descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

NVOD_REFERENCE
public static final short **NVOD_REFERENCE**

The constant indicates the descriptor tag value of the NVOD reference service descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

PARENTAL_RATING
public static final short **PARENTAL_RATING**

The constant indicates the descriptor tag value of the parental rating descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

PARTIAL_RECEPTION
public static final short **PARTIAL_RECEPTION**

The constant indicates the descriptor tag value of the partial reception descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

PARTIAL_TRANSPORT_STREAM
public static final short **PARTIAL_TRANSPORT_STREAM**

The constant indicates the descriptor tag value of the partial transport stream descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

PARTIALTS_TIME
public static final short **PARTIALTS_TIME**

The constant indicates the descriptor tag value of the partial transport stream time descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

REFERENCE
public static final short **REFERENCE**

The constant indicates the descriptor tag value of the reference descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

SATELLITE_DELIVERY_SYSTEM
public static final short **SATELLITE_DELIVERY_SYSTEM**

The constant indicates the descriptor tag value of the satellite delivery system descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

SERIES
public static final short **SERIES**

The constant indicates the descriptor tag value of the series descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

SERVICE
public static final short **SERVICE**

The constant indicates the descriptor tag value of the service descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

SERVICE_LIST

public static final short **SERVICE_LIST**

The constant indicates the descriptor tag value of the service list descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

SHORT_EVENT

public static final short **SHORT_EVENT**

The constant indicates the descriptor tag value of the short form event descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

SHORT_NODE_INFORMATION

public static final short **SHORT_NODE_INFORMATION**

The constant indicates the descriptor tag value of the short form node information descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

SI_PARAMETER

public static final short **SI_PARAMETER**

The constant indicates the descriptor tag value of the SI transmission parameter descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

SI_PRIME_TS

public static final short **SI_PRIME_TS**

The constant indicates the descriptor tag value of the SI prime TS descriptor specified in ARIB STD-B10.

Reference:

Chapter 16 List of Constant Value

STC_REFERENCE
public static final short **STC_REFERENCE**

The constant indicates the descriptor tag value of the STC reference descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

STREAM_IDENTIFIER
public static final short **STREAM_IDENTIFIER**
The constant indicates the descriptor tag value of the stream identification descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

STUFFING
public static final short **STUFFING**

The constant indicates the descriptor tag value of the stuffing descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

SYSTEM_MANAGEMENT
public static final short **SYSTEM_MANAGEMENT**

The constant indicates the descriptor tag value of the system management descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

TARGET_AREA
public static final short **TARGET_AREA**

The constant indicates the descriptor tag value of the target area descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

TERRESTRIAL_DELIVERY_SYSTEM
public static final short **TERRESTRIAL_DELIVERY_SYSTEM**

The constant indicates the descriptor tag value of the terrestrial delivery system descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

TIME_SHIFTED_EVENT
public static final short **TIME_SHIFTED_EVENT**

The constant indicates the descriptor tag value of the time shifted event descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

TIME_SHIFTED_SERVICE
public static final short **TIME_SHIFTED_SERVICE**

The constant indicates the descriptor tag value of the time shifted service descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

TS_INFORMATION
public static final short **TS_INFORMATION**

The constant indicates the descriptor tag value of the TS information descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

VIDEO_DECODE_CONTROL
public static final short **VIDEO_DECODE_CONTROL**

The constant indicates the descriptor tag value of the video decode control descriptor specified in ARIB STD-B10.

Reference:
Chapter 16 List of Constant Value

M.2.3 PMTElementaryStream
jp.or.arib.tv.si
Interface PMTElementaryStream

Super-interface for all:
SIInformation

public interface **PMTElementaryStream**
extends SIInformation

The interface indicates the elementary stream of the service (channel).

In each service, PMT is present to describe the elementary stream of the service. This means that the object mounted with the interface indicates one of such elementary streams. Each object mounted with the PMTElementaryStream interface is identified by the combination of original_network_id, transport_stream_id, service_id and component_tag (or elementary_PID).

Reference:

PMTService, PMTStreamType

General information on the field

Field inherited from the interface jp.or.arib.tv.si.SIInformation
FROM_CACHE_ONLY, FROM_CACHE_OR_STREAM, FROM_STREAM_ONLY

General information on the method	
ARIBLocator	**getARIBLocator** () To acquire ARIBLocator that identifies the elementary stream.
int	**getComponentTag** () To acquire the component tag.
short	**getElementaryPID** () To acquire elementary_PID.
int	**getOriginalNetworkID** () To acquire the value of the original network ID.
int	**getServiceID** () To acquire the value of the service ID.
byte	**getStreamType** () To acquire the stream type ID of the elementary stream.
int	**getTransportStreamID** () To acquire the value of the transport stream ID.

Method inherited from jp.or.arib.tv.si.SIInformation
fromActual, getDataSource, getDescriptorTags, getSIDatabase, getUpdateTime, retrieveDescriptors, retrieveDescriptors

Details of the method

getARIBLocator
public ARIBLocator **getARIBLocator**()

To acquire the ARIBLocator that identifies the elementary stream.

Return value:

ARIBLocator of the elementary stream

getComponentTag
public int **getComponentTag**()

To acquire the component tag

Return value:

Component tag. If the component tag has not been assigned to correspond the elementary stream, "-2" is returned.

getElementaryPID
public short **getElementaryPID**()

To acquire elementary_PID

Return value:

PID of the transport stream that transmits the data of the elementary stream

getOriginalNetworkID
public int **getOriginalNetworkID**()

To acquire the value of the original network ID

Return value:

Original network ID

getServiceID
public int **getServiceID**()

To acquire the value of the service ID

Return value:

Service ID

getStreamType
public byte **getStreamType**()

To acquire the stream type ID of the elementary stream. The value to be returned is the actual value in the descriptor loop and not necessarily the value defined by PMTStreamType.

Return value:

Stream type ID (The part of the return value is the one defined by PMTElementaryStream interface.)

Reference:

PMTStreamType

getTransportStreamID
public int **getTransportStreamID**()
To acquire the value of the transport stream ID
Return value:
Transport stream ID

M.2.4 PMTService
jp.or.arib.tv.si
Interface PMTService

Super-interface for all:
SIInformation

public interface **PMTService**
extends SIInformation

This interface indicates the specific service to be transmitted by the transport stream. The information was acquired from PMT.

Each object mounted with PMTService interface is identified by the combination of original_network_id, transport_stream_id, and service_id PMTService.

General information on the field

Field inherited from the Interface jp.or.arib.tv.si.SIInformation
FROM_CACHE_ONLY, FROM_CACHE_OR_STREAM, FROM_STREAM_ONLY

General information on the method	
ARIBLocator	**getARIBLocator**() To acquire ARIBLocator that identifies this service.
int	**getOriginalNetworkID** () To acquire the original network ID.
int	**getPcrPid**() To acquire PID of PCR.
int	**getServiceID**() To acquire the service ID.

int	**getTransportStreamID**() To acquire the transport stream ID.
SIRequest	**retrievePMTElementaryStreams**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] somePMTDescriptorTags) To acquire the information relevant to the elementary streams that compose this service from PMT.

Method inherited from the Interface jp.or.arib.tv.si.SIInformation
fromActual, getDataSource, getDescriptorTags, getSIDatabase, getUpdateTime, retrieveDescriptors, retrieveDescriptors

Details of the method

getARIBLocator

public ARIBLocator **getARIBLocator**()

To acquire the ARIBLocator that identifies this service

Return value:

ARIBLocator of this service

getOriginalNetworkID

public int **getOriginalNetworkID**()

To acquire the original network ID

Return value:

Original network ID

getPcrPid

public int **getPcrPid**()

To acquire PID of PCR

Return value:

PID of PCR

getServiceID

public int **getServiceID**()

To acquire the service ID

Return value:

Service ID

getTransportStreamID

public int **getTransportStreamID**()

To acquire the transport stream ID

Return value:
Transport stream ID

retrievePMTElementaryStreams
public SIRequest **retrievePMTElementaryStreams** (short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
short[] somePMTDescriptorTags)
throws SIIllegalArgumentException

The information relevant to the elementary stream composing this service is acquired from PMT. In the SIIterator to be returned with the normal end event of the requests, one or more objects mounted with PMTElementaryStream interface are included. If there is no corresponding object, the appropriate event among the following events is notified:

- SIObjectNotInCacheEvent
- SIObjectNotInTableEvent
- SITableNotFoundEvent

This method acquires PMTElementaryStream from the sub-table of the same version number as the one of the PMTService instance. If the sub-table of the version number is unusable, SITableUpdatedEvent is returned.

Parameter:
retrieveMode - specifies the mode for data acquisition.
Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (only from the stream).

appData – Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need the data, “null” can be assigned.

listener – SIRetrievalListener to receive the event of request completion notification.

somePMTDescriptorTags – Hint list of descriptors requested by the application (the descriptor is identified by the tag value). In case there is one element of array and the value is “-1”. this means the application requests all descriptors. In case somePMTDescriptorTags is “null,” this means the application does not need any of the descriptors. An inappropriate tag value is ignored.

Return value:
SIRequest object
Exception:
SIIllegalArgumentException – In case the retrieveMode is an invalid value.
Reference:
SIRequest, SIRetrievalListener, PMTElementaryStream

M.2.5 PMTStreamType
jp.or.arib.tv.si
Interface PMTStreamType

public interface **PMTStreamType**
This interface defines constants that correspond to various stream types.

Reference:
PMTElementaryStream, PMTElementaryStream.getStreamType ()

General information on the field	
static byte	**DSMCC_DATA_CAROUSEL** The constant indicates the stream type of DSM-CC data carousel defined by ISO/IEC 13818-1.
static byte	**INDEPENDENT_PES** The constant indicates the stream type of independent PES defined by ISO/IEC 13818-1.
static byte	**MPEG1_AUDIO** The constant indicates the audio stream type of MPEG1 defined by ISO/IEC 13818-1.
static byte	**MPEG1_VIDEO** The constant indicates the video stream type of MPEG1 defined by ISO/IEC 13818-1.
static byte	**MPEG2_AAC_AUDIO** This constant indicates the audio stream type of MPEG2AAC defined by ISO/IEC 13818-1.
static byte	**MPEG2_AUDIO** The constant indicates the audio stream type of MPEG2 defined by ISO/IEC 13818-1.
static byte	**MPEG2_VIDEO** This constant indicates the video stream type of MPEG2 defined by ISO/IEC 13818-1.
static byte	**MPEG4_VIDEO** The constant indicates the video stream type of MPEG4 defined by ISO/IEC 13818-1.
static byte	**MPEG4_AVC_VIDEO** This constant indicates the video stream type of H.264/MPEG-4 AVC defined by ISO/IEC 13818-1.

Details of the field

DSMCC_DATA_CAROUSEL

public static final byte **DSMCC_DATA_CAROUSEL**

This constant indicates the stream type of DSM-CC data carousel defined by ISO/IEC 13818-1.

Reference:

Chapter 16 List of Constant Value

INDEPENDENT_PES

public static final byte **INDEPENDENT_PES**

The constant indicates the stream type of independent PRS defined by ISO/IEC 13818-1.

Reference:

Chapter 16 List of Constant Value

MPEG1_AUDIO

public static final byte **MPEG1_AUDIO**

The constant indicates the audio stream type of MPEG1 defined by ISO/IEC 13818-1.

Reference:

Chapter 16 List of Constant Value

MPEG1_VIDEO

public static final byte **MPEG1_VIDEO**

The constant indicates the video stream type of MPEG1 defined by ISO/IEC 13818-1.

Reference:

Chapter 16 List of Constant Value

MPEG2_AAC_AUDIO

public static final byte **MPEG2_AAC_AUDIO**

This constant indicates the audio stream type of MPEG2AAC defined by ISO/IEC 13818-1.

Reference:

Chapter 16 List of Constant Value

MPEG2_AUDIO

public static final byte **MPEG2_AUDIO**

The constant indicates the audio stream type of MPEG2 defined by ISO/IEC 13818-1.

Reference:

Chapter 16 List of Constant Value

MPEG2_VIDEO
public static final byte **MPEG2_VIDEO**

This constant indicates the video stream type of MPEG2 defined by ISO/IEC 13818-1.

Reference:

Chapter 16 List of Constant Value

MPEG4_VIDEO
public static final byte **MPEG4_VIDEO**

The constant indicates the video stream type of MPEG4 defined by ISO/IEC 13818-1.

Reference:

Chapter 16 List of Constant Value

MPEG4_AVC_VIDEO
public static final byte **MPEG4_AVC_VIDEO**

This constant indicates the video stream type of H.264/MPEG-4 AVC defined by ISO/IEC 13818-1.

Reference:

Chapter 16 List of Constant Value

M.2.6 SIBouquet
jp.or.arib.tv.si
Interface SIBouquet

Super-interface for all:

SIInformation

public interface **SIBouquet**
extends SIInformation

This interface indicates the sub-table of Bouquet Association Table (BAT) that described the specific bouquet (together with the SITransportStreamBAT).
Each object mounting SIBouquet interface is identified by bouquet_id identifier.

Reference:

SITransportStreamBAT

General information on the field

Field inherited from Interface jp.or.arib.tv.si.SIInformation

FROM_CACHE_ONLY, FROM_CACHE_OR_STREAM, FROM_STREAM_ONLY

General information of the method	
int	**getBouquetID**() To acquire the bouquet ID.
short[]	**getDescriptorTags**() This method defines additional semantics to SIInformation#getDescriptorTags method.
java.lang.String	**getName**() This method returns the bouquet name to be described in the bouquet descriptor.
ARIBLocator[]	**getSIServiceLocators**() This method is to acquire the list of the ARIBLocator to identify the service that belongs to the service.
SIRequest	**retrieveDescriptors**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener) This method defines the additional semantics to the first prototype of SIInformation#retrieveDescriptors method.
SIRequest	**retrieveDescriptors**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) This method defines the additional semantics to the second prototype of SIInformation#retrieveDescriptors method.
SIRequest	**retrieveSIBouquetTransportStreams**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) This method acquires information relevant to the transport stream to which the bouquet belongs.

Method inherited from Interface jp.or.arib.tv.si.SIInformation
fromActual, getDataSource, getSIDatabase, getUpdateTime

Details of the method

getBouquetID

public int **getBouquetID**()

To acquire the bouquet ID

Return value:

The bouquet ID of this bouquet

getDescriptorTags

public short[] **getDescriptorTags**()

This method defines the additional semantics to SIInformation#getDescriptorTags method. If the BAT sub-table to be the original of this SIBouquet object is composed of multiple sections, the descriptor tags are returned in the order of the appearance at the

time when the descriptor loop of each section is connected.

Definition:

getDescriptorTags in the Interface SIInformation

Return value:

The descriptor tag of the descriptor being actually broadcast and corresponding to the object (to be identified by the tag)

Reference:

SIInformation, SIInformation.getDescriptorTags()

getName

public java.lang.String **getName**()

This method returns the bouquet name of the bouquet to be described in the bouquet name descriptor. If the information is not usable, "" is returned. All control codes defined in ARIB STD-B10 are ignored except for SP, APR and APD. Each character of 8bit-code used in ARIB-SI is converted into an appropriate Unicode character.

Return value:

Bouquet name of this bouquet

getSIServiceLocators

public ARIBLocator[] **getSIServiceLocators**()

To acquire the list of the ARIBLocator to identify the service that belongs to the bouquet.

Return value:

Array of the ARIBLocator to identify the service

Reference:

ARIBLocator, SIService

retrieveDescriptors

public SIRequest **retrieveDescriptors**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener)
throws SIIllegalArgumentException

This method defines the additional semantics to the first prototype of SIInformation#retrieveDescriptors method. If the BAT sub-table to be the original of this SIBouquet object is composed of multiple sections, the descriptor tags are returned in the order of the appearance at the time when the descriptor loop of each section is connected.

Definition:

retrieveDescriptors in the Interface SIInformation

Parameter:

retrieveMode – specifies the data acquisition mode.
Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (only from the stream).
appData – Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need the data, "null" is assigned.
listener - SIRetrievalListener to receive the event of request completion notification.
Return value:
SIRequest object
Exception:
SIIllegalArgumentException - In case the retrieveMode is an invalid value.
Reference:
SIInformation, SIInformation.retrieveDescriptors (short, java.lang. Object, jp.or.arib.tv.si.SIRetrievalListener)

retrieveDescriptors
public SIRequest **retrieveDescriptors** (short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This method defines the additional semantics to the second prototype of SIInformation#retrieveDescriptors. If the BAT sub-table to be the original of this SIBouquet object is composed of multiple sections, the descriptor tags are returned in the order of the appearance at the time when the descriptor loop of each section is connected.
Definition:
retrieveDescriptors in the Interface SIInformation
Parameter:
retrieveMode – specifies the data acquisition mode.

Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (only from the stream).

appData – Object is to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need the data, "null" is assigned.

listener - SIRetrievalListener to receive the event of request completion notification.

someDescriptorTags - Hint list of descriptors requested by the application (the descriptor is identified by the tag value). In case there is one element of array and the value is "-1," this means the application requests all descriptors. If someDescriptorTags indicates "null," this means the application does not need any of the descriptors. Any value that is not in the effective range of the descriptor (0-255) is ignored except in the case where the array element is one and the value is "-1".

Return value:
SIRequest object
Exception:
SIIllegalArgumentException - In case the retrieveMode is an invalid value.
Reference:
SIInformation, SIInformation.retrieveDescriptors (short, Object, SIRetrievalListener, short[])

retrieveSIBouquetTransportStreams
public SIRequest **retrieveSIBouquetTransportStreams** (short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
short[] someDescriptorTags)
throws SIIllegalArgumentException

To acquire the information relevant to the transport stream to which the bouquet belongs.

In the SIIterator to be returned with the normal end event of the requests, one or more objects mounted with PMTElementaryStream Interface is included. This method acquires SITransportStreamBAT from the sub-table of the same version number as SIBouquet instance. When the sub-table of the version number is not usable, SITableUpdateEvent is returned.

Parameter:
retrieveMode – specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (only from the stream).

appData - Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need the data, "null" is assigned.

listener - SIRetrievalListener to receive the event of request completion notification.

someDescriptorTags - Hint list of descriptors requested by the application (the descriptor is identified by the tag value). In case there is one element of array and the value is "-1," this means the application requests all descriptors. If someDescriptorTags indicate "null," this means the application does not need any of the descriptors. Any value that is not in the effective range of the descriptor (0-255) is ignored except in the case where the array element is one and the value is "-1".

Return value:
SIRequest object
Exception:
SIIllegalArgumentException - In case the retrieveMode is an invalid value.
Reference:
SIRequest, SIRetrievalListener, SITransportStreamBAT, DescriptorTag

M.2.7 SIBroadcaster
jp.or.arib.tv.si
Interface SIBroadcaster

Super-interface for all:
SIInformation

public interface **SIBroadcaster**
extends SIInformation

This interface indicates the specific broadcaster in the service. The information to be acquired by the interface method is to be acquired from BIT table.

Each object mounted with the SIBroadcaster interface is identified by broadcaster_id identifier.

General information on the field

Field inherited from Interface jp.or.arib.tv.si.SIInformation
FROM_CACHE_ONLY, FROM_CACHE_OR_STREAM, FROM_STREAM_ONLY

General information on the method	
int	**getBroadcasterID** () This method is to acquire the broadcaster ID.

boolean	**getBroadcastViewProperty** () This method is to acquire the value of the broadcaster's display propriety.
java.lang.String	**getName** () This method returns the broadcaster name of the broadcaster to be described in the broadcaster descriptor.
int	**getOriginalNetworkID**() This method is to acquire the original network ID.
ARIBLocator[]	**getSIServiceLocators**() This method is to acquire the list of the ARIBLocator to identify the service that belongs to the broadcaster.
SIRequest	**retrieveOriginalNetworkDescriptors**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener) This method is to acquire all descriptors to be transmitted by the first loop of BIT.
SIRequest	**retrieveOriginalNetworkDescriptors**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) This method is to acquire the descriptor collection to be transmitted by the first loop of BIT.

Method inherited from Interface jp.or.arib.tv.si.SIInformation
fromActual, getDataSource, getDescriptorTags, getSIDatabase, getUpdateTime, retrieveDescriptors, retrieveDescriptors

Details of the method

getBroadcasterID

public int **getBroadcasterID**()

This method is to acquire the broadcaster ID.

Return value:

The value of the broadcaster ID

getBroadcastViewProperty

public boolean **getBroadcastViewProperty**()

This method is to acquire the value of the broadcaster's display propriety. If the user presentation based on the unit as defined by the broadcaster name is available, it is true.

Return value:

If the value of the broadcasting company's display propriety is "1," it is true. If not, it is false.

getName

public java.lang.String **getName**()

This method returns the broadcaster name of the broadcaster that is described in the broadcaster name descriptor. If the descriptor is not being transmitted, "" is returned. All

control codes defined in ARIB STD-B10 are ignored except for SP, APR and APD. Each character of 8bit-code used in ARIB-SI is converted into an appropriate Unicode character.

Return value:

Broadcaster name

getOriginalNetworkID

public int **getOriginalNetworkID**()

To acquire the original network ID.

Return value:

The value of the original network ID.

getSIServiceLocators

public ARIBLocator[] **getSIServiceLocators**()

To acquire the list of the ARIBLocator to identify the service that belongs to the broadcaster.

Return value:

Array of the ARIBLocator to identify the service

Reference:

ARIBLocator, SIService

retrieveOriginalNetworkDescriptors

public SIRequest **retrieveOriginalNetworkDescriptors**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener)
throws SIIllegalArgumentException

This method acquires all descriptors to be transmitted by the first loop of BIT. The descriptors are acquired in the order of the transmission by the loop.

This method is nonsynchronous and the completion is clarified by notifying SISuccessfulRetrieveEvent to the listener. The acquired descriptors are included in the SIIterator that is returned by getResult method of the event. If the descriptors are present (in the loop), the Descriptor object is included in the SIIterator and if the descriptors are absent, nothing is included.

Parameter:

retrieveMode – specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (only from the stream).

appData - Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need the data, “null” is assigned.

listener - SIRetrievalListener to receive the event of request completion notification.

Return value:

SIRequest object

Exception:

SIIllegalArgumentException - In case the retrieveMode is an invalid value.

Reference:

SIRequest, SIRetrievalListener, Descriptor, SIIterator

retrieveOriginalNetworkDescriptors

public SIRequest **retrieveOriginalNetworkDescriptors**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This method acquires the collection of the descriptors to be transmitted by the first loop of BIT. The descriptors are acquired in the order of the transmission by the loop. The value in the parameter someDescriptorTags is used for selection of the descriptor to be acquired. Only the descriptors that have the tag value in the array of someDescriporTags and someDescriptorTags, and if there is one array element and the value is "-1," all descriptors relevant to the object is acquired.

If this tag value list can indicate substratum mounting (in case of object request calling method), (additional) cache organization can have an effect.

This method is nonsynchronous and the completion is clarified by notifying SISuccessfulRetrieveEvent to the listener. The acquired descriptors are included in the SIIterator that is returned by the getResult method of the event. If the descriptors are present (in the loop), the Descriptor object is included in the SIIterator and if the descriptors are absent, nothing is included.

Parameter:

retrieveMode – Specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (only from the stream).

appData - Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need the data, "null" is assigned.

listener - SIRetrievalListener to receive the event of request completion notification.

someDescriptorTags - The array of descriptor tags of descriptors. This is to be used for selection of the descriptors to be included SI table to correspond to this SIBroadcaster. If there is one element of array and the value is "-1," all descriptors is acquired.

Return value:

SIRequest object

Exception:
SIIllegalArgumentException - In case the retrieveMode is an invalid value.
Reference:
SIRequest, SIRetrievalListener, Descriptor, SIIterator, DescriptorTag

M.2.8 SIEvent
jp.or.arib.tv.si
Interface SIEvent

Super-interface for all:
SIInformation

public interface **SIEvent**
extends SIInformation

This interface indicates the specific program in the service.

Each object mounted with the SIevent interface is identified by the combination of original_network_id identifier, transport_stream_id, service_id and event_id.

If the return value of the method is acquired from the short form event descriptor and more than one descriptor is present, the following algorithm must be used.

In case the language returned by javax.tv.service.SIManager#getPreferredLanguage method is used for the short form event descriptor, the value is returned from the descriptor. If not, it depends on the mounting status that is to be used out of the available short form event descriptors.

Reference:
SIService

General information on the field

Field inherited from Interface jp.or.arib.tv.si.SIInformation
FROM_CACHE_ONLY, FROM_CACHE_OR_STREAM, FROM_STREAM_ONLY

General information on the method	
ARIBLocator	**getARIBLocator**() To acquire the ARIBLocator to identify the program.
java.lang.String[]	**getAudioComponentDescriptions**() This method returns the character description of audio elementary stream relevant to the program.

java.lang.String[]	**getComponentDescriptions**() This method returns the character description of elementary stream relevant to the program.
byte[]	**getContentNibbles**() This method returns the genre of the program.
java.lang.String[]	**getDataContentDescriptions**() This method returns the character description relevant to the data broadcasting program.
long	**getDuration**() This method acquires the program duration.
int	**getEventID**() This method acquires the event ID.
SIExEventInformation[]	**getExEventInformations**() This method returns detailed information relevant to the program.
boolean	**getFreeCAMode** () This method acquires the scramble value of the program.
byte[]	**getLevel1ContentNibbles**() This method returns the first step classification of the content ID in the program.
java.lang.String	**getName**() This method returns the name of the program.
int	**getOriginalNetworkID**() This method acquires the original network ID.
byte	**getRunningStatus**() This method acquires the running status of the program.
java.lang.String	**getSeriesName**() This method returns the name of series relevant to the program.
int	**getServiceID**() This method acquires the service ID.
java.lang.String	**getShortDescription**() This method returns the program description of the program.
java.util.Date	**getStartTime**() This method acquires the start time of the program.
int	**getTransportStreamID**() This method acquires the transport stream ID.
byte[]	**getUserNibbles**() This method returns the user genre relevant to the program.
SIRequest	**retrieveSIService**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) This method acquires SIService that indicates the service. The service means the one to which the program indicated by the SIEvent belongs.

Method inherited from Interface jp.or.arib.tv.si.SIInformation

fromActual, getDataSource, getDescriptorTags, getSIDatabase, getUpdateTime, retrieveDescriptors, retrieveDescriptors

Details of the method

getARIBLocator

public ARIBLocator **getARIBLocator**()

This method acquires the ARIBLocator that identifies the program.

Return value:

ARIBLocator of this program

getAudioComponentDescriptions

public java.lang.String[] **getAudioComponentDescriptions**()

This method returns the character description of the audio elementary stream relevant to the program. The information can be acquired from the audio component describer. If there is no descriptor, an empty array (the array length is “0”) is returned. The return value is the array and one audio elementary stream is described for each element.

Return value:

Character description of the audio elementary stream relevant to the program.

getComponentDescriptions

public java.lang.String[] **getComponentDescriptions**()

This method returns the character description of the elementary stream relevant to the program. This information can be acquired from the component descriptor. If there is no descriptor, an empty array (the array length is “0”) is returned. The return value is the array and one elementary stream is described for each element.

Return value:

Character description of the elementary stream relevant to the program

getContentNibbles

public byte[] **getContentNibbles**()

This method returns the genre of the program. This information can be acquired from the content descriptor. If there is no descriptor, an empty array (the array length is “0”) is returned. The return value is the array and one genre is described for each element. The top 4 bits are for genre 1 (the first stage classification) and the bottom 4 bits are for the second stage classification.

Return value:

Genre relevant to the program. The top 4 bits are for genre 1 (the first stage classification) and the bottom 4 bits are for the second stage classification.

getDataContentDescriptions
public java.lang.String[] **getDataContentDescriptions**()

This method returns the character description relevant to the data broadcast program. This data can be acquired from the data contents descriptor. If the describer is not present, an empty array (the array length is "0") is returned. The return value is the array and individual contents of the data broadcast program are described by element.

Return value:
Character description relevant to the contents of the data broadcast program

getDuration
public long **getDuration**()

This method acquires the duration of the program.

Return value:
Duration (sec.)

getEventID
public int **getEventID**()

This method acquires the event ID.

Return value:
Event ID

getExEventInformations
public SIExEventInformation[] **getExEventInformations**()

This method returns the detailed information relevant to the program. SIExEventInformation object to be returned can be acquired from the extended event descriptor. If more than one item is present, objects are generated for each item and allocated on the array in the order of appearance. If it is composed of more than one extended event descriptors, they are also allocated on the array in the order of appearance when these descriptors are connected. If there is no descriptor, an empty array (the array length is "0") is returned. The character description stored in the extended description cannot be acquired by this method.

Return value:
Array of SIExEventInformation objects

getFreeCAMode
public boolean **getFreeCAMode**()

This method acquires the scramble value of the program. If it is false, this indicates all components of the program are not scrambled.

Return value:

Scramble value

getLevel1ContentNibbles
public byte[] **getLevel1ContentNibbles**()

This method returns the first stage classification of the content ID in the program. This information can be acquired from the content descriptor. If there is no descriptor, an empty array (the array length is "0") is returned. The return value is the array and one genre is described for each element. As for the data of each genre, the bottom 4 bits are bytes to be returned the top 4 bits are set for "0".

Return value:

The first stage classification in the program genre

getName
public java.lang.String **getName**()

This method returns the name of the program. The program name can be acquired from the short form event descriptor. If the information is not usable, "" is returned. All control codes defined in ARIB STD-B10 are ignored except for SP, APR and APD. Each character of 8bit-code used in ARIB-SI is converted into an appropriate Unicode character.

Return value:

The program name of the program

getOriginalNetworkID
public int **getOriginalNetworkID**()

This method acquires the original network ID.

Return value:

Original network ID

getRunningStatus
public byte **getRunningStatus**()

This method acquires the running status of the program.

Return value:

Running status (SIRunningStatus defines the possible value to be acquired.)

Reference:

SIRunningStatus

getSeriesName
public java.lang.String **getSeriesName**()

This method returns the series name relevant to the program. This information can be acquired from the series descriptor. If there is no descriptor, null object is returned.

Return value:
Series name relevant to the program

getServiceID
public int **getServiceID**()

This method acquires the service ID.

Return value:
Service ID

getShortDescription
public java.lang.String **getShortDescription**()

This method returns the program description of the program. The program description can be acquired from the short form event descriptor. If the information is not usable, "" is returned. Each character of 8bit-code used in ARIB-SI is converted into an appropriate Unicode character.

Return value:
The program description of the program (short form)

getStartTime
public java.util.Date **getStartTime**()

This method acquires the start time of the program.

Return value:
The start time of the program

getTransportStreamID
public int **getTransportStreamID**()

This method acquires the transport stream ID.

Return value:
Transport stream ID

getUserNibbles
public byte[] **getUserNibbles**()

This method returns the user genre relevant to the program. This information can be acquired form the content descriptor. If the describer is not present, an empty array (the array length is "0") is returned. The return value is the array and one genre is described for each element.

Return value:
User genre relevant to the program

retrieveSIService

public SIRequest **retrieveSIService**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This is the method to acquire SIService that indicates the service. The service means the one to which the program indicated by the SIEvent belongs.

SIIterator to be returned with the normal end event of the request includes more than one object mounted with the SIService interface. If there is no corresponding object, the appropriate event among the ones shown below is notified:

- SIObjectNotInCacheEvent
- SIObjectNotInTableEvent
- SITableNotFoundEvent

Parameter:

retrieveMode – specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (only from the stream).

appData - Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need the data, “null” is assigned.

listener - SIRetrievalListener to receive the event of request completion notification.

someDescriptorTags - Hint list of descriptors requested by the application (the descriptor is identified by the tag value). In case there is one element of array and the value is “-1,” this means the application requests all descriptors. If someDescriptorTags indicate “null,” this means the application does not need any of the descriptors. Any value that is not in the effective range of the descriptor (0-255) is ignored except in the case where the array element is one and the value is “-1”.

Return value:

SIRequest object

Exception:

SIIllegalArgumentException – In case the retrieveMode is an invalid value.

Reference:

SIRequest, SIRetrievalListener, SIService, DescriptorTag

M.2.9 SIInformation
jp.or.arib.tv.si
Interface SIInformation

Known sub-interface list:

PMTElementaryStream, PMTService, SIBouquet, SIBroadcaster, SIEvent, SINetwork, SIService, SITime, SITransportStream, SITransportStreamBAT, SITransportStreamNIT

public interface **SIInformation**

This interface is a collection of common functions of SIBouquet, SIBroadcaster, SINetwork, SITransportStream, SIService, PMTService, SIEvent, SITime and PMTElementaryStream. Each SIInformation interface indicates the sub-table (or part of the sub-table). Each method that accesses the descriptors acquires the descriptors from the sub-table of the same version number as the SIInformation instance. If the version number is not usable, SITableUpdatedEvent is returned.

Reference:

SIBouquet, SIBroadcaster, SINetwork, SITransportStream, SIService, PMTService, SIEvent, SITime, PMTElementaryStream

General information on the field	
static short	**FROM_CACHE_ONLY** The constant is used for the acquisition mode parameter of acquisition method.
static short	**FROM_CACHE_OR_STREAM** The constant is used for the acquisition mode parameter of acquisition method.
static short	**FROM_STREAM_ONLY** The constant is used for the acquisition mode parameter of acquisition method.

General information on the method	
boolean	**fromActual**() If the information included in the object mounted with this interface was selected from the “actual” table or the table that does not distinguish “actual” or “not actual,” “true” is returned.
org.davic.mpeg.TransportStream	**getDataSource**() This method returns the org.davic.mpeg.TransportStream object that was selected from the information included in the object mounted with this interface.
short[]	**getDescriptorTags**() This method acquires the tag values of all descriptors that comprise part of the current version of this object.

SIDatabase	**getSIDatabase**() This method returns the root of the hierarchical structure to which the object mounted with this interface belongs.
java.util.Date	**getUpdateTime**() This method returns the last updated day and time of this information included in the object mounted with this interface.
SIRequest	**RetrieveDescriptors** (short retrieveMode, java.lang.Object appData, SIRetrievalListener listener) This method acquires all descriptors in the order of broadcasting.
SIRequest	**RetrieveDescriptors** (short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) This method acquires descriptors partially.

Details of the field

FROM_CACHE_ONLY

public static final short **FROM_CACHE_ONLY**

This is the constant to be used for the parameter of the method acquisition mode. When FROM_CACHE_ONLY is designated, the data is acquired only from the cache. If it is not acquirable, SINotInCacheEvent is sent to the listener. In this case, there is no access to the stream.

Reference:

Chapter 16 List of Constant Value

FROM_CACHE_OR_STREAM

public static final short **FROM_CACHE_OR_STREAM**

This is the constant to be used for the parameter of the method acquisition mode. When FROM_CACHE_OR_STREAM is designated, the cache is acquired if the data is in the cache. If not, the data is acquired from the stream.

Reference:

Chapter 16 List of Constant Value

FROM_STREAM_ONLY

public static final short **FROM_STREAM_ONLY**

This is the constant to be used for the parameter of the method acquisition mode. When FROM_CACHE_OR_STREAM is designated, the data is acquired from the stream directly and there is no access to the cache from the beginning. This mode is useful only when the application is aware that data is not in the cache or the data in the cache is invalid. However, it is not necessary to think about the availability of cache data at the SI database mounting. If the presence of data with an updated version number was informed by the SI API listener mechanism, this means that the update information has already been sent to the mounted SI database. In this case, the application should adopt

FROM_CACHE_OR_STREAM mode. If the updated data has already been cached, the data acquisition speed could be much higher.

Reference:

Chapter 16 List of Constant Value

Details of the method

fromActual

public boolean **fromActual**()

If the information included in the object mounted with this interface was selected from the "actual" table or the table does not distinguish "actual" or "not actual," "true" is returned.

Return value:

If the information was selected from the "actual" table or the table does not distinguish "actual" or "not actual," then "true" is returned. If not, "false" is returned.

getDataSource

public org.davic.mpeg.TransportStream **getDataSource**()

This method returns the org.davic.mpeg.TransportStream object that was selected from the information included in the object mounted with this interface.

Return value:

org.davic.mpeg.TransportStream Object selected from the information

Reference:

TransportStream

getDescriptorTags

public short[] **getDescriptorTags**()

This method acquires the tag values of all descriptors that comprise part of the current version of this object. This method returns the tag value of the descriptor but does not hint whether the data is present in the cache. It is not a necessary function for the method either. If there is no descriptor that is associated with this SI Information, empty array with "0" array length is returned.

Return value:

The tag of the descriptor for an object that is actually on-aired

Reference:

DescriptorTag

getSIDatabase

public SIDatabase **getSIDatabase**()

This method returns the root of the hierarchical structure to which the object mounted

with this interface belongs.

Return value:

The root of the hierarchical structure

getUpdateTime

public java.util.Date **getUpdateTime**()

This method returns the last updated day and time of this information included in the object mounted with this interface.

Return value:

The last updated day and time

retrieveDescriptors

public SIRequest **retrieveDescriptors** (short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener)
throws SIIllegalArgumentException

This method acquires all descriptors in the order of broadcasting.

This method is nonsynchronous. When it is completed, SISuccessfulRetrieveEvent is sent to the listener. The acquired descriptor is included in the SIIterator, which is returned by the getResult method of the event. Presence of the descriptor indicates that the Iterator includes the Descriptor object. Absence of the corresponding descriptor indicates that the Iterator does not include the object.

Parameter:

retrieveMode – designates the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (only from the stream).

appData - Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need the data, “null” is assigned.

listener - SIRetrievalListener to receive the event of request completion notification.

Return value:

SIRequest object

Exception:

SIIllegalArgumentException - In case the retrieveMode is an invalid value.

Reference:

SIRequest, SIRetrievalListener, Descriptor, SIIterator

retrieveDescriptors

public SIRequest **retrieveDescriptors**(short retrieveMode,
java.lang.Object appData,

SIRetrievalListener listener,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This method acquires descriptors partially. In the order of broadcasting, all descriptors or some of the descriptors are acquired.

The tag value included in the parameter someDescriptorTags is used for the selection of the descriptors to be returned. Only the descriptors that have the tag values included in the someDescriptorTags array. If the element of the array is one and the value is "-1," all descriptors associated with the object are acquired.

If this tag value list can hint substratum mounting (in case of the object request calling method), (additional) cache organization can have an effect.

This method is nonsynchronous and the completion is clarified by notifying the SISuccessfulRetrieveEvent of the listener. The acquired descriptors are included in the SIIterator that is returned by the getResult method of the event. If the descriptors are present (in the loop), the Descriptor object is included in the Iterator and if corresponding descriptors are absent, no object is included in the Iterator.

Parameter:
retrieveMode – specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (only from the stream).
appData - Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need the data, "null" is assigned.
listener - SIRetrievalListener to receive the event of request completion notification.
someDescriptorTags – An array of the descriptor tags to be used for descriptor selection included in the SI table that corresponds to this SIInformation object. If the element of the array is one and the value is "-1," all descriptors are acquired.
Return value:
SIRequest object
Exception:
SIIllegalArgumentException - In case the retrieveMode is an invalid value.
Reference:
SIRequest, SIRetrievalListener, Descriptor, SIIterator, DescriptorTag

M.2.10 SIIterator
jp.or.arib.tv.si

Interface SIIterator

Super-interface for all:
java.util.Enumeration

public interface **SIIterator**
extends java.util.Enumeration

The object mounted with the SIIterator interface can access the contents via the collection of SI objects. In order to keep the consistency in the object collection, some accesses to the stream are not initialized depending on access to the contents.

General information on the method	
int	**numberOfRemainingObjects**() The number of objects that are maintained in the iterator

Method inherited from Interface java.util.Enumeration
hasMoreElements, nextElement

Details of the method

numberOfRemainingObjects
public int **numberOfRemainingObjects**()
The number of objects that are maintained in the iterator
Return value:
The number of objects that are maintained

M.2.11 SIMonitoringListener
jp.or.arib.tv.si
Interface SIMonitoringListener

Super-interface for all:
java.util.EventListener

public interface **SIMonitoringListener**
extends java.util.EventListener

Interface mounted with application class in order to receive the changes of monitoring the SI object.

Reference:

SIMonitoringEvent

General information on the method	
void	**postMonitoringEvent**(SIMonitoringEvent anEvent) This method is called back by the SI API mounted so as to inform the event listener.

Details of the method

postMonitoringEvent

public void **postMonitoringEvent**(SIMonitoringEvent anEvent)

This method is called back by the SI API mounted so as to inform the event listener.

Parameter:

anEvent – Event to be informed

Reference:

SIMonitoringEvent

M.2.12 SIMonitoringType

jp.or.arib.tv.si

Interface SIMonitoringType

public interface **SIMonitoringType**

This interface defines the constant that corresponds to each type of SI information in the SIMonitoringEvent.

Reference:

SIMonitoringListener, SIMonitoringEvent

General information on the field	
static byte	**BOUQUET** Constant for the SIInformation object to indicate the bouquet
static byte	**BROADCASTER** Constant for the SIInformation object to indicate the broadcaster
static byte	**NETWORK** Constant for the SIInformation object to indicate the network
static byte	**PMT_SERVICE** Constant for the SIInformation object to indicate the PMT service
static byte	**PRESENT_FOLLOWING_EVENT** Constant for the SIInformation object to indicate the EIT [Present/Following]
static byte	**SCHEDULED_EVENT** Constant for the SIInformation object to indicate the EIT [Schedule]

static byte	**SERVICE** Constant for the SIInformation object to indicate the service

Details of the field

BOUQUET

public static final byte **BOUQUET**

Constant for the SIInformation object to indicate the bouquet

Reference:

Chapter 16 List of Constant Value

BROADCASTER

public static final byte **BROADCASTER**

Constant for the SIInformation object to indicate the broadcaster

Reference:

Chapter 16 List of Constant Value

NETWORK

public static final byte **NETWORK**

Constant for the SIInformation object to indicate the network

Reference:

Chapter 16 List of Constant Value

PMT_SERVICE

public static final byte **PMT_SERVICE**

Constant for the SIInformation object to indicate the PMT service

Reference:

Chapter 16 List of Constant Value

PRESENT_FOLLOWING_EVENT

public static final byte **PRESENT_FOLLOWING_EVENT**

Constant for the SIInformation object to indicate the EIT [Present/Following]

Reference:

Chapter 16 List of Constant Value

SCHEDULED_EVENT

public static final byte **SCHEDULED_EVENT**

Constant for the SIInformation object to indicate the EIT [Schedule]

Reference:

Chapter 16 List of Constant Value

SERVICE
public static final byte **SERVICE**

Constant for the SIInformation object to indicate the service

Reference:

Chapter 16 List of Constant Value

M.2.13 SINetwork
jp.or.arib.tv.si
Interface SINetwork

Super-interface for all:

SIInformation

public interface **SINetwork**
extends SIInformation

This interface indicates the sub-table of the Network Information Table (NIT) that describes a specific network (together with the SITransportStreamNIT).

Each object that is mounted with the SINetwork interface is identified by the network_id identifier.

Reference:

SITransportStream, SITransportStreamNIT

General information on the field

Field inherited from Interface jp.or.arib.tv.si.SIInformation
FROM_CACHE_ONLY, FROM_CACHE_OR_STREAM, FROM_STREAM_ONLY

General information on the method	
short[]	**getDescriptorTags**() This method defines the additional semantics to SIInformation#getDescriptorTags method.
java.lang.String	**getName**() This method returns the network name of the network that is described in the network name describer.
int	**getNetworkID**() This method acquires the network ID of this network.

SIRequest	**RetrieveDescriptors**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener) This method defines the additional semantics to the first prototype of the SIInformation#retrieveDescriptors method.
SIRequest	**retrieveDescriptors**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) This method defines the additional semantics to the second prototype of the SIInformation#retrieveDescriptors method.
SIRequest	**retrieveSITransportStreams**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) This method acquires the information about the transport stream to be transmitted via the network.

Method inherited from the Interface jp.or.arib.tv.si.SIInformation
fromActual, getDataSource, getSIDatabase, getUpdateTime

Details of the method

getDescriptorTags

public short[] **getDescriptorTags**()

This method defines the additional semantics to the SIInformation#getDescriptorTags method. If the NIT sub-table as an original of the SIInformation#getDescriptorTags method is comprised of multiple sections, the description tags are returned in the order of the connection of the descriptor loop of each section.

Definition:

getDescriptorTags in the interface SIInformation

Return value:

Descriptor tag of the actually on-aired descriptor corresponding to the object (to be identified by a tag)

Reference:

SIInformation, SIInformation.getDescriptorTags()

getName

public java.lang.String **getName**()

This method returns the network name of the network that is described in the network name describer. If the information is unusable, " " is returned. All control codes defined by ARIB STD-B10 except for SP, APR and APD are ignored. Each character of the 8bit-code in ARIB-SI is converted into an appropriate Unicode indication.

Return value:

Network ID of the network

getNetworkID
public int **getNetworkID**()

This method acquires the network ID of this network.

Return value:

Network ID of the network

retrieveDescriptors
public SIRequest **retrieveDescriptors**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener)
throws SIIllegalArgumentException

This method defines the additional semantics to the first prototype of the SIInformation#retrieveDescriptors method. If NIT sub-table as an original of the SIInformation#getDescriptorTags method is comprised of multiple sections, the description tags are returned in the order of the connection of the descriptor loop of each section.

Definition:

retrieveDescriptors in the interface SIInformation

Parameter:

retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (only from the stream).

appData - Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need the data, “null” is assigned.

listener - SIRetrievalListener to receive the event of request completion notification.

Return value:

SIRequest object

Exception:

SIIllegalArgumentException - In case the retrieveMode is an invalid value.

Reference:

SIInformation, SIInformation.retrieveDescriptors (short, java.lang.Object, jp.or.arib.tv.si.SIRetrievalListener)

retrieveDescriptors
public SIRequest **retrieveDescriptors**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
short[] someDescriptorTags)

throws SIIllegalArgumentException

This method defines the additional semantics to the second prototype of the SIInformation#retrieveDescriptors method. If NIT sub-table as an original of the SIInformation#getDescriptorTags method is comprised of multiple sections, the description tags are returned in the order of the connection of the descriptor loop of each section.

Definition

retrieveDescriptors in the interface SIInformation

Parameter:

retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (only from the stream).

appData - Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need the data, “null” is assigned.

listener - SIRetrievalListener to receive the event of request completion notification.

someDescriptorTags - Hint list of descriptors requested by the application (the descriptor is identified by the tag value). In case there is one element of array and the value is “-1,” this means the application requests all descriptors. If someDescriptorTags indicate “null,” this means the application does not need any of the descriptors. Any value that is not in the effective range of the descriptor (0-255) is ignored except in the case where the array element is one and the value is “-1”.

Return value:

SIRequest object

Exception:

SIIllegalArgumentException - In case retrieveMode is an invalid value.

Reference:

SIInformation, SIInformation.retrieveDescriptors (short, Object, SIRetrievalListener, short[])

retrieveSITransportStreams

public SIRequest **retrieveSITransportStreams**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This method acquires the information relevant to the transport stream that is transmitted via the network.

In the SIIterator to be returned with the normal end event of the requests, one or more objects mounted with the SITransportStreamNIT Interface is included. This method acquires SITransportStream from the sub-table of the same version number as the SINetwork instance. When the sub-table of the version number is not usable, SITableUpdateEvent is returned.

Parameter:
retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (only from the stream).
appData - Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need the data, "null" is assigned.
listener - SIRetrievalListener to receive the event of request completion notification.

someDescriptorTags - Hint list of descriptors requested by the application (the descriptor is identified by the tag value). In case there is one element of array and the value is "-1," this means the application requests all descriptors. If someDescriptorTags indicate "null," this means the application does not need any of the descriptors. Any value that is not in the effective range of the descriptor (0-255) is ignored, unless the array element is one and the value is "-1".

Return value:
SIRequest object
Exception:
SIIllegalArgumentException - In case the retrieveMode is an invalid value.
Reference:
SIRequest, SIRetrievalListener, SITransportStreamNIT, DescriptorTag

M.2.14 SIRetrievalListener
jp.or.arib.tv.si
Interface SIRetrievalListener

Super-interface for all:
java.util.EventListener

public interface **SIRetrievalListener**
extends java.util.EventListener

The application must mount this interface so as to receive the completion event of the SI

request.

Reference:

SIRetrievalEvent

General information on the method	
void	**postRetrievalEvent**(SIRetrievalEvent event) This method is called from the SI API mounted so as to notify the SI request completion of the listener.

Details of the method

postRetrievalEvent

public void **postRetrievalEvent**(SIRetrievalEvent event)

This method is called from the SI API mounted so as to notify the SI request completion of the listener.

Parameter:

event – Event object

Reference:

SIRetrievalEvent

M.2.15 SIRunningStatus

jp.or.arib.tv.si

Interface SIRunningStatus

public interface **SIRunningStatus**

This interface defines the constant that corresponds to the value of running status in the service and event.

General information on the field	
static byte	**NOT_RUNNING** The constant, as defined in ARIB STD-B10, indicates that the running status is “not running”.
static byte	**PAUSING** The constant, as defined in ARIB STD-B10, indicates that the running status is “pausing”.
static byte	**RUNNING** The constant, as defined in ARIB STD-B10, indicates that the running status is “running”.
static byte	**STARTS_IN_A_FEW_SECONDS** The constant, as defined in ARIB STD-B10, indicates that the running status is ready to “start in a few seconds”.
static	**UNDEFINED**

byte	The constant, as defined in ARIB STD-B10, indicates that the running status is “undefined”.

Details of the field

NOT_RUNNING

public static final byte **NOT_RUNNING**

The constant, as defined in ARIB STD-B10, indicates that the running status is “not running”.

Reference:

Chapter 16 List of Constant Value

PAUSING

public static final byte **PAUSING**

The constant, as defined in ARIB STD-B10, indicates that the running status is “pausing”.

Reference:

Chapter 16 List of Constant Value

RUNNING

public static final byte **RUNNING**

The constant, as defined in ARIB STD-B10, indicates that the running status is “running”.

Reference:

Chapter 16 List of Constant Value

STARTS_IN_A_FEW_SECONDS

public static final byte **STARTS_IN_A_FEW_SECONDS**

The constant, as defined in ARIB STD-B10, indicates that the running status is ready to “start in a few seconds”.

Reference:

Chapter 16 List of Constant Value

UNDEFINED

public static final byte **UNDEFINED**

The constant, as defined in ARIB STD-B10, indicates that the running status is “undefined”.

Reference:

Chapter 16 List of Constant Value

M.2.16 SIService
jp.or.arib.tv.si
Interface SIService

Super-interface for all:
SIInformation

public interface **SIService**
extends SIInformation

This interface indicates a specific service that is transmitted by one transport stream. The information acquired via the method of this interface is acquired from SDT.
Each object mounted with the SIService interface is identified by combination of the following Ids:
Original network ID, Transport stream ID, Service ID

General information on the field

Field inherited from Interface jp.or.arib.tv.si.SIInformation
FROM_CACHE_ONLY, FROM_CACHE_OR_STREAM, FROM_STREAM_ONLY

General information on the method	
ARIBLocator	**getARIBLocator**() This method acquires the ARIBLocator to identify this service.
boolean	**getEITPresentFollowingFlag**() This method acquires EIT_present_following_flag value.
boolean	**getEITScheduleFlag**() This method acquires EIT_schedule_flag value.
int	**getEITUserDefinedFlag**() This method acquires EIT_user_defined_flags value.
boolean	**getFreeCAMode**() This method acquires free_CA_mode value.
java.lang.String	**getName**() This method returns the name that indicates that the service included in the service_descriptor.
int	**getOriginalNetworkID**() This method acquires the original network ID.
java.lang.String	**getProviderName**() This method returns the name of the service provider included in the service descriptor.
byte	**getRunningStatus**()

	This method acquires the running status of this service.
int	**getServiceID**() This method acquires the service ID.
short	**getSIServiceType**() This method acquires the service type.
int	**getTransportStreamID**() This method acquires the transport stream ID.
SIRequest	**retrieveFollowingSIEvent**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) This method acquires the information relevant to the following program from EIT[Present/Following].
SIRequest	**retrievePMTService**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) This method acquires the PMTService information relevant to this service.
SIRequest	**retrievePresentSIEvent**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) This method acquires the information relevant to the present program from EIT[Present/Following].
SIRequest	**retrieveScheduledSIEvents**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags, java.util.Date startTime, java.util.Date endTime) This method acquires the information relevant to the program scheduled in the designated period from EIT[Schedule].

Method inherited from Interface jp.or.arib.tv.si.SIInformation
fromActual, getDataSource, getDescriptorTags, getSIDatabase, getUpdateTime, retrieveDescriptors, retrieveDescriptors

Details of the method

getARIBLocator

public ARIBLocator **getARIBLocator**()

This method acquires the ARIBLocator to identify this service.

Return value:

ARIBLocator of this service

getEITPresentFollowingFlag

public boolean **getEITPresentFollowingFlag**()

This method acquires EIT_present_following_flag value. The presence of EIT [Present/Following] information of this service is indicated by “true”.

Return value:

EIT_present_following_flag value

getEITScheduleFlag
public boolean **getEITScheduleFlag**()

This method acquires EIT_schedule_flag value. The presence of EIT[Schedule] information of this service is indicated by “true”.

Return value:
EIT_schedule_flag value

getEITUserDefinedFlag
public int **getEITUserDefinedFlag**()

This method acquires EIT_user_defined_flags value.

Return value:
EIT_user_defined_flags value

getFreeCAMode
public boolean **getFreeCAMode**()

This method acquires free_CA_mode value. In case of “false,” no components are scrambled in this service.

Return value:
free_CA_mode value of this service

getName
public java.lang.String **getName**()

This method returns the name that indicates that the service included in the service descriptor. If there is no descriptor, ““ is returned. All control codes defined in ARIB STD-B10 are ignored except for SP, APR and APD. Each character of 8bit-code used in ARIB-SI is converted into an appropriate Unicode character.

Return value:
Service name

getOriginalNetworkID
public int **getOriginalNetworkID**()

This method acquires the original network ID.

Return value:
Original network ID

getProviderName
public java.lang.String **getProviderName**()

This method returns the name of the service provider included in the service descriptor. If there is no descriptor, empty array ““ is returned. All control codes defined in ARIB STD-B10 are ignored except for SP, APR and APD. Each character of 8bit-code used in

ARIB-SI is converted into an appropriate Unicode character.

Return value:

The name of this service provider

getRunningStatus

public byte **getRunningStatus**()

This method acquires the running status of this service.

Return value:

Running status. The acquirable value is defined by the SIRunningStatus interface.

Reference:

SIRunningStatus

getServiceID

public int **getServiceID**()

This method acquires the service ID.

Return value:

Service ID

getSIServiceType

public short **getSIServiceType**()

This method acquires the service type. The service type is picked up from the service descriptor.

Return value:

Service type. The acquirable value is defined by the SIService Type interface.

Reference:

SIServiceType

getTransportStreamID

public int **getTransportStreamID**()

This method acquires the transport stream ID.

Return value:

Transport stream ID

retrieveFollowingSIEvent

public SIRequest **retrieveFollowingSIEvent**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This method acquires the information relevant to the following program from

EIT[Present/Following].

SIIterator to be returned from the event that notifies the normal end of the request includes one object mounted with an SIEvent interface. If there is no appropriate object, a suitable event among the ones shown below is notified.

SIObjectNotInCacheEvent, SIObjectNotInTableEvent, SITableNotFoundEvent

Parameter:
retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (only from the stream).
appData – Optional object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, “null” is assigned.
listener - SIRetrievalListener to receive the event of request completion notification.
someDescriptorTags - Tag list of the descriptors that the application concerns. If the application concerns all descriptors, only one element of value “-1” is included in the array. If the application does not concern any of the descriptors, “null” is assigned for someDescriptorTags. Any value that is out of the proper range (0-255) as a descriptor is ignored except in case of “-1,” which is a single element having a special meaning.

Return value:
SIRequest object
Exception:
SIIllegalArgumentException - In case the retrieveMode is false.
Reference:
SIRequest, SIRetrievalListener, SIEvent, DescriptorTag

retrievePMTService
public SIRequest **retrievePMTService**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
short[] someDescriptorTags)
throws SIIllegalArgumentException
This method acquires the PMTService relevant to this service.

SIIterator to be returned form the event that notifies the normal end of the request stores one object mounted with PMTService interface. If there is no appropriate object, a suitable event among the ones shown below is notified.

SIObjectNotInCacheEvent, SIObjectNotInTableEvent, SITableNotFoundEvent

Parameter:
retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (only from the stream).
appData – Optional object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, "null" is assigned.
listener - SIRetrievalListener to receive the event of request completion notification.
someDescriptorTags - Tag list of the descriptors that the application concerns. If the application concerns all descriptors, only one element of value "-1" is included in the array. If the application does not concern any of the descriptors, "null" is assigned for someDescriptorTags. Any value that is out of the proper range (0-255) as a descriptor is ignored except in case of "-1," which is a single element having a special meaning.

Return value:
SIRequest object
Exception:
SIIllegalArgumentException – In case the retrieveMode is false.
Reference:
SIRequest, SIRetrievalListener, PMTService, DescriptorTag

retrievePresentSIEvent
public SIRequest **retrievePresentSIEvent**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
short[] someDescriptorTags)
throws SIIllegalArgumentException
This method acquires the information relevant to the present program from EIT[Present/Following].

SIIterator to be returned from the event that notifies the normal end of the request stores one object mounted with PMTEvent interface. If there is no appropriate object, a suitable event among the ones shown below is notified.

SIObjectNotInCacheEvent, SIObjectNotInTableEvent, SITableNotFoundEvent

Parameter:

retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (only from the stream).

appData – Optional object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, "null" is assigned.

listener - SIRetrievalListener to receive the event of request completion notification.

someDescriptorTags - Tag list of the descriptors that the application concerns. If the application concerns all descriptors, only one element of value "-1" is included in the array. If the application does not concern any of the descriptors, "null" is assigned for someDescriptorTags. Any value that is out of the proper range (0-255) as a descriptor is ignored except in the case of "-1," which is a single element having a special meaning.

Return value:

SIRequest Object

Exception:

SIIllegalArgumentException – In case the retrieveMode is false.

Reference:

SIRequest, SIRetrievalListener, SIEvent, DescriptorTag

retrieveScheduledSIEvents

public SIRequest **retrieveScheduledSIEvents**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
short[] someDescriptorTags,
java.util.Date startTime,
java.util.Date endTime)
throws SIIllegalArgumentException,
SIInvalidPeriodException

This method acquires the information relevant to the present program scheduled in the designated period from EIT [Schedule].

Programs are present in the same order as in EIT [Schedule].

SIIterator to be returned from the event that notifies the normal end of the request stores more than one object that is mounted with the SIEvent interface.

Parameter:

retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY

(only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (only from the stream).
appData – Optional object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, "null" is assigned.
listener - SIRetrievalListener to receive the event of request completion notification.
someDescriptorTags - Tag list of the descriptors that the application concerns. If the application concerns all descriptors, only one element of value "-1" is included in the array. If the application does not concern any of the descriptors, "null" is assigned for someDescriptorTags. Any value that is out of the proper range (0-255) as a descriptor is ignored except in the case of "-1," which is a single element having a special meaning.
startTime – Start time of requested period
endTime – End time of requested period
Return value:
SIRequest object
Exception:
SIIllegalArgumentException – In case the retrieveMode is false.
SIInvalidPeriodException – In case the designated period is false.
Reference:
SIRequest, SIRetrievalListener, SIEvent, DescriptorTag

M.2.17 SIServiceType
jp.or.arib.tv.si
Interface SIServiceType

public interface **SIServiceType**

Definition of service type

Reference:
SIService.getSIServiceType()

General information on the field	
static short	**BOOKMARK_LIST** The constant, as defined in ARIB STD-B10, indicates that the service type is a "bookmark list data service".
static short	**DATA** The constant, as defined in ARIB STD-B10, indicates that the service type is a "data service".
static	**DATA_EXCLUSIVE_FOR_ACCUMULATION**

short	The constant, as defined in ARIB STD-B10, indicates that the service type is a “data service exclusive for accumulation”.
static short	**DATA_FOR_ACCUMULATION_IN_ADVANCE** The constant, as defined in ARIB STD-B10, indicates that the service type is a “data service exclusive for accumulation in advance”.
static short	**DIGITAL_AUDIO** The constant, as defined in ARIB STD-B10, indicates that the service type is a “digital audio service”.
static short	**DIGITAL_TELEVISION** The constant, as defined in ARIB STD-B10, indicates that the service type is a “digital TV service”.
static short	**ENGINEERING_DOWNLOAD** The constant, as defined in ARIB STD-B10, indicates that the service type is an “engineering download service”.
static short	**PROMOTION_DATA** The constant, as defined in ARIB STD-B10, indicates that the service type is a “promotion data service”.
static short	**PROMOTION_SOUND** The constant, as defined in ARIB STD-B10, indicates that the service type is a “promotion sound service”.
static short	**PROMOTION_VIDEO** The constant, as defined in ARIB STD-B10, indicates that the service type is a “promotion video service”.
static short	**SPECIAL_AUDIO** The constant, as defined in ARIB STD-B10, indicates that the service type is a “special audio service”.
static short	**SPECIAL_DATA** The constant, as defined in ARIB STD-B10, indicates that the service type is a “special data service”.
static short	**SPECIAL_VIDEO** The constant, as defined in ARIB STD-B10, indicates that the service type is a “special video service”.
static short	**UNKNOWN** The constant, as defined in ARIB STD-B10, indicates that the service type is an “unknown”.

Details of the field

BOOKMARK_LIST

public static final short **BOOKMARK_LIST**

The constant, as defined in ARIB STD-B10, indicates that the service type is a “bookmark list data service”.

Reference:

Chapter 16 List of Constant Value

DATA

public static final short **DATA**

The constant, as defined in ARIB STD-B10, indicates that the service type is a “data service”.

Reference:

Chapter 16 List of Constant Value

DATA_EXCLUSIVE_FOR_ACCUMULATION

public static final short **DATA_EXCLUSIVE_FOR_ACCUMULATION**

The constant, as defined in ARIB STD-B10, indicates that the service type is a “data service exclusive for accumulation”.

Reference:

Chapter 16 List of Constant Value

DATA_FOR_ACCUMULATION_IN_ADVANCE

public static final short **DATA_FOR_ACCUMULATION_IN_ADVANCE**

The constant, as defined in ARIB STD-B10, indicates that the service type is a “data service exclusive for accumulation in advance”.

Reference:

Chapter 16 List of Constant Value

DIGITAL_AUDIO

public static final short **DIGITAL_AUDIO**

The constant, as defined in ARIB STD-B10, indicates that the service type is a “digital audio service”.

Reference:

Chapter 16 List of Constant Value

DIGITAL_TELEVISION

public static final short **DIGITAL_TELEVISION**

The constant, as defined in ARIB STD-B10, indicates that the service type is a “digital TV service”.

Reference:

Chapter 16 List of Constant Value

ENGINEERING_DOWNLOAD

public static final short **ENGINEERING_DOWNLOAD**

The constant, as defined in ARIB STD-B10, indicates that the service type is an “engineering download service”.

Reference:

Chapter 16 List of Constant Value

PROMOTION_DATA
public static final short **PROMOTION_DATA**

The constant, as defined in ARIB STD-B10, indicates that the service type is a "promotion data service".

Reference:

Chapter 16 List of Constant Value

PROMOTION_SOUND
public static final short **PROMOTION_SOUND**

The constant, as defined in ARIB STD-B10, indicates that the service type is a "promotion sound service".

Reference:

Chapter 16 List of Constant Value

PROMOTION_VIDEO
public static final short **PROMOTION_VIDEO**

The constant, as defined in ARIB STD-B10, indicates that the service type is a "promotion video service".

Reference:

Chapter 16 List of Constant Value

SPECIAL_AUDIO
public static final short **SPECIAL_AUDIO**
The constant, as defined in ARIB STD-B10, indicates that the service type is a "special audio service".

Reference:

Chapter 16 List of Constant Value

SPECIAL_DATA
public static final short **SPECIAL_DATA**
The constant, as defined in ARIB STD-B10, indicates that the service type is a "special data service".

Reference:

Chapter 16 List of Constant Value

SPECIAL_VIDEO
public static final short **SPECIAL_VIDEO**
The constant, as defined in ARIB STD-B10, indicates that the service type is a “special video service”.

Reference:
Chapter 16 List of Constant Value

UNKNOWN
public static final short **UNKNOWN**
The constant, as defined in ARIB STD-B10, indicates that the service type is an “unknown”.

Reference:
Chapter 16 List of Constant Value

M.2.18 SITime
jp.or.arib.tv.si
Interface SITime

Super-interface for all:
SIInformation

public interface **SITime**
extends SIInformation

This interface indicates time and date table (TDT). If the object indicates TDT, the behavior of the retrieveDescriptors method and the getDescriptorTags method are the same as the description in the case that there is no descriptor (because TDT has no descriptor).

Reference:
SIDatabase

General information on the field

Field inherited from Interface jp.or.arib.tv.si.SIInformation
FROM_CACHE_ONLY, FROM_CACHE_OR_STREAM, FROM_STREAM_ONLY

General information on the method

java.util.Date	**getTime**() This method acquires the coded time in TDT or TOT.

Method inherited from jp.or.arib.tv.si.SIInformation
fromActual, getDataSource, getDescriptorTags, getSIDatabase, getUpdateTime, retrieveDescriptors, retrieveDescriptors

Details of the method

getTime

public java.util.Date **getTime**()

This method acquires the coded time in TDT or TOT. These tables transmits the time by JST but note that the Date type object to be acquired by this method does not include a time zone concept. (Internal indication is UTC base).

Return value:

Time

Reference:

Date

M.2.19 SITransportStream

jp.or.arib.tv.si

Interface SITransportStream

Super-interface for all:

SIInformation

Known sub-interface list:

SITransportStreamBAT, SITransportStreamNIT

public interface **SITransportStream**
extends SIInformation

Base interface to indicate the information relevant to the transport stream.
The method to acquire the transport stream in the SIDatabase class and SINetwork interface returns the object mounted with the SITransportStreamNIT interface referring NIT.
The method to acquire the transport stream in the SIBouqet interface returns the object mounted with the SITransportStreamBAT interface referring BAT.

General information on the field

Field inherited from Interface jp.or.arib.tv.si.SIInformation
FROM_CACHE_ONLY, FROM_CACHE_OR_STREAM, FROM_STREAM_ONLY

General information on the method	
ARIBLocator	**getARIBLocator**() This method acquires the ARIBLocator that identifies the transport stream.
int	**getOriginalNetworkID**() This method acquires original network ID.
int	**getTransportStreamID**() This method acquires transport stream ID.
SIRequest	**retrieveSIServices**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) This method acquires information relevant to the service to be transmitted by the transport stream.

Method inherited from Interface jp.or.arib.tv.si.SIInformation
fromActual, getDataSource, getDescriptorTags, getSIDatabase, getUpdateTime, retrieveDescriptors, retrieveDescriptors

Details of the method

getARIBLocator

public ARIBLocator **getARIBLocator**()

This method acquires the ARIBLocator that identifies the transport stream.

Return value:

ARIBLocator that indicates this transport stream

getOriginalNetworkID

public int **getOriginalNetworkID**()

This method acquires original network ID.

Return value:

Original network ID

getTransportStreamID

public int **getTransportStreamID**()

This method acquires transport stream ID.

Return value:

Transport stream ID

retrieveSIServices
public SIRequest **retrieveSIServices**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This method acquires information relevant to the service to be transmitted by the transport stream. This method operates to the object mounted with the SITransportStreamNIT interface and SITransportStreamBAT interface in the same manner.

SIIterator to be returned from the event that notifies the normal end of the request stores the object mounted with the SIService interface.

Parameter:
retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (only from the stream).
appData – Optional object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, “null” is assigned.
listener - SIRetrievalListener to receive the event of request completion notification.
someDescriptorTags - Tag list of the descriptors that the application concerns. If the application concerns all descriptors, only one element of value “-1” is included in the array. If the application does not concern any of the descriptors, “null” is assigned for someDescriptorTags. Any value that is out of the proper range (0-255) as a descriptor is ignored except in the case of “-1,” which is a single element having a special meaning.

Return value:
SIRequest object
Exception:
SIIllegalArgumentException - In case the retrieveMode is an invalid value.
Reference:
SIRequest, SIRetrievalListener, SIService, DescriptorTag

M.2.20 SITransportStreamBAT
jp.or.arib.tv.si
Interface SITransportStreamBAT

Super-interface for all:

SIInformation, SITransportStream

public interface **SITransportStreamBAT**
extends SITransportStream

This interface indicates information relevant to the transport stream acquired form BAT. All methods that access descriptors return the descriptor information acquired from BAT. The method to acquire the transport stream in the SIBouquet returns the object mounted with this interface.

General information on the field

Field inherited from Interface jp.or.arib.tv.si.SIInformation
FROM_CACHE_ONLY, FROM_CACHE_OR_STREAM, FROM_STREAM_ONLY

General information on the method	
int	**getBouquetID**() The method acquires the ID of the bouquet to which this transport stream belongs.

Method inherited from Interface jp.or.arib.tv.si.SITransportStream
getARIBLocator, getOriginalNetworkID, getTransportStreamID, retrieveSIServices

Method inherited from Interface jp.or.arib.tv.si.SIInformation
fromActual, getDataSource, getDescriptorTags, getSIDatabase, getUpdateTime, retrieveDescriptors, retrieveDescriptors

Details of the method

getBouquetID
public int **getBouquetID**()

The method acquires the bouquet ID to which this transport stream belongs.

Return value:

Bouquet ID

M.2.21 SITransportStreamNIT
jp.or.arib.tv.si
Interface SITransportStreamNIT

Super-interface for all:

SIInformation, SITransportStream

public interface **SITransportStreamNIT**
extends SITransportStream

The interface indicates information relevant to the transport steam acquired from NIT. All methods that access descriptors return the descriptor information acquired from NIT. The method to acquire the transport stream in the SIDatabase or SINetwork returns the object mounted with this interface.

General information on the field

Field inherited from Interface jp.or.arib.tv.si.SIInformation
FROM_CACHE_ONLY, FROM_CACHE_OR_STREAM, FROM_STREAM_ONLY

General information on the method	
int	**getNetworkID**() This method acquires the ID of the network to which this transport stream belongs.

Method inherited from Interface jp.or.arib.tv.si.SITransportStream
getARIBLocator, getOriginalNetworkID, getTransportStreamID, retrieveSIServices

Method inherited from Interface jp.or.arib.tv.si.SIInformation
fromActual, getDataSource, getDescriptorTags, getSIDatabase, getUpdateTime, retrieveDescriptors, retrieveDescriptors

Details of the method

getNetworkID
public int **getNetworkID**()

This method acquires the ID of the network to which this transport stream belongs.

Return value:

Network ID

M.2.22 Descriptor
jp.or.arib.tv.si
Class Descriptor

java.lang.Object
  |
  +--**jp.or.arib.tv.si.Descriptor**

public class **Descriptor**
extends java.lang.Object

This class indicates the descriptors of the sub-table.
A descriptor consists of three fields as follows.

- Descriptor tag
- Descriptor length
- Descriptor data

The descriptor tag uniquely identifies the type of each descriptor and the descriptor length indicates the bytes of the data section.

The data section is a byte array expressed by the size of the descriptor length. The contents depend on the descriptor type.

Reference:

DescriptorTag

General information on the method	
byte	**getByteAt**(int index) This method acquires the specific byte value of the descriptor data section.
byte[]	**getContent**() This method acquires a copy of the descriptor data section (whole section behind the descriptor length field).
Short	**getContentLength**() This method returns the length of the data section indicated in the descriptor length field.
short	**getTag**() This method acquires the descriptor tag.

Method inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Details of the method

getByteAt

public byte **getByteAt**(int index)
throws java.lang.IndexOutOfBoundsException

This method acquires the specific byte value of the descriptor data section.

Parameter:

index – index corresponding to the data section. The value "0" corresponds to the initial bytes immediately after the descriptor length data.

Return value:

Requested byte

Exception:

java.lang.IndexOutOfBoundsException – In case of index < 0 or index > = ContentLength

getContent

public byte[] **getContent**()

This method acquires a copy of the descriptor data section (whole section behind the descriptor length field).

Return value:

Copy of the descriptor data section

getContentLength

public short **getContentLength**()

This method returns the length of the data section indicated in the descriptor length field.

Return value:

The length of the data section

getTag

public short **getTag**()

This method acquires the descriptor tag. The value to be returned is the one that is actually used and not necessarily the one defined by DescriptorTag.

Return value:

Descriptor tag (Generally the value defined by DesriptorTag interface)

Reference:

DescriptorTag

M.2.23 SIDatabase
jp.or.arib.tv.si
Class SIDatabase

java.lang.Object
  |
  +--**jp.or.arib.tv.si.SIDatabase**

public class **SIDatabase**
extends java.lang.Object

This class indicates the root of the SI information hierarchical structure. There is one SIDatabase for each network interface. Therefore, there is only one SIDatabase if there is only one network interface.

General information of field	
static int	**RETRIEVE_ALL_INFORMATIONS**
static int	**RETRIEVE_CURRENT_SELECTED**

General information on the method	
void	**addBouquetMonitoringListener**(SIMonitoringListener listener, int bouquetId) This method initializes the monitoring operation of bouquet information.
void	**addBroadcasterMonitoringListener**(SIMonitoringListener listener, int broadcasterId) This method initializes the monitoring operation of broadcaster information.
void	**addEventPresentFollowingMonitoringListener**(SIMonitoringListener listener, int originalNetworkId, int transportStreamId, int serviceId) This method initializes the monitoring operation of EIT [Present/Following] information.
void	**addEventScheduleMonitoringListener**(SIMonitoringListener listener, int originalNetworkId, int transportStreamId, int serviceId, java.util.Date startTime, java.util.Date endTime) This method initializes the monitoring operation of EIT [Schedule] information.
void	**addNetworkMonitoringListener**(SIMonitoringListener listener, int networkId) This method initializes the monitoring operation of network information.
void	**addPMTServiceMonitoringListener**(SIMonitoringListener listener, int originalNetworkId, int transportStreamId, int serviceId) This method initializes the monitoring operation of PMT information relevant to the service.
void	**addServiceMonitoringListener**(SIMonitoringListener listener, int originalNetworkId, int transportStreamId) This method initializes the monitoring operation of SDT information relevant to the

	service.
static SIDatabase[]	**getSIDatabase**() This method returns the array of the SIDatabase object (for each network interface).
void	**removeBouquetMonitoringListener**(SIMonitoringListener listener, int bouquetId) This method deletes the event listener registry for bouquet information monitoring.
void	**removeBroadcasterMonitoringListener**(SIMonitoringListener listener, int broadcasterId) This method removes the event listener registry for broadcaster information monitoring.
void	**removeEventPresentFollowingMonitoringListener**(SIMonitoringListener listener, int originalNetworkId, int transportStreamId, int serviceId) This method removes the event listener registry for EIT[Present/Following] information monitoring.
void	**removeEventScheduleMonitoringListener**(SIMonitoringListener listener, int originalNetworkId, int transportStreamId, int serviceId) Corresponding to the whole scheduled span, this method removes the event listener registry for EIT [Schedule] information monitoring.
void	**removeNetworkMonitoringListener**(SIMonitoringListener listener, int networkId) This method removes the event listener registry for network information monitoring.
void	**removePMTServiceMonitoringListener**(SIMonitoringListener listener, int originalNetworkId, int transportStreamId, int serviceId) This method removes the event listener registry for PMT information monitoring relevant to the service.
void	**removeServiceMonitoringListener**(SIMonitoringListener listener, int originalNetworkId, int transportStreamId) This method removes the event listener registry for information monitoring relevant to the service.
SIRequest	**retrieveActualSINetwork**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) This method acquires information relevant to the present network.
SIRequest	**retrieveActualSIServices**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) This method acquires information relevant to the actual service.
SIRequest	**retrieveActualSITransportStreaM (**short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) This method acquires information relevant to the actual transport stream.
SIRequest	**retrievePMTElementaryStreams**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, ARIBLocator aribLocator, short[] someDescriptorTags) This method acquires PMT elementary stream information relevant to the service component from the actual transport stream of this SIDatabase object.
SIRequest	**retrievePMTElementaryStreams**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, int serviceId, int componentTag, short[] someDescriptorTags) This method acquires PMT elementary stream information relevant to the service component from the actual transport stream of this SIDatabase object.

SIRequest	**retrievePMTService**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, ARIBLocator aribLocator, short[] someDescriptorTags) This method acquires PMT information relevant to the service.
SIRequest	**retrievePMTServices**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, int serviceId, short[] someDescriptorTags) This method acquires PMT information relevant to the service from the actual transport stream of this SIDatabase object.
SIRequest	**retrieveSIBouquets**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, int bouquetId, short[] someDescriptorTags) This method acquires information relevant to the bouquet.
SIRequest	**retrieveSIBroadcaster** (short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, intoriginalNetworkId, int broadcasterId, short [] some DescriptorTags) This method acquires information relevant to the broadcaster.
SIRequest	**retrieveSIBroadcasters** (short retrieveMode, java.lang. Object appData, SIRetrievalListener listener, int originalNetworkId, short [] some DescriptorTags) This method acquires information relevant to the broadcaster specified by originalNetworkId.
SIRequest	**retrieveSINetworks**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, int networkId, short[] someDescriptorTags) This method acquires information relevant to the network.
SIRequest	**retrieveSIService**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, ARIBLocator aribLocator, short[] someDescriptorTags) This method acquires information relevant to the service.
SIRequest	**retrieveSIServices**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, int originalNetworkId, int transportStreamId, int serviceId, short[] someDescriptorTags) This method acquires information relevant to the service.
SIRequest	**retrieveSITimeFromTDT**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener) This method acquires information relevant to the time from Time Date Table (TDT) on the actual transport stream.
SIRequest	**retrieveSITimeFromTOT**(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) This method acquires information relevant to the time from Time Offset Table (TOT) on the actual transport stream.

Method inherited from the class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Details of field

RETRIEVE_ALL_INFORMATIONS

public static final int **RETRIEVE_ALL_INFORMATIONS**

Reference:

Chapter 16 List of Constant Value

RETRIEVE_CURRENT_SELECTED
public static final int **RETRIEVE_CURRENT_SELECTED**
Reference:
Chapter 16 List of Constant Value

Details of the method

addBouquetMonitoringListener
public void **addBouquetMonitoringListener**(SIMonitoringListener listener, int bouquetId)
throws SIIllegalArgumentException

This method initializes the monitoring operation. When bouquet information is changed, the registered listener object is notified of the event.

The monitoring operation depends on the mounting status. Continued monitoring is not necessarily required. The event of change is notified as soon as possible even if delayed compared with the actual change on the stream due to the resources distribution for the monitoring operation scheduled between tables. This standard does not specify the minimum requirements of the SI table monitoring operation. The mounted system should be optimum but depends on the status. However, when an application embedded with the receiver, or the ARIB application, for example, detects the change at SI information acquisition from the stream, only the notification of the change to the listener is required.

When the network interface relevant to the SIDatabase object starts to select the station for another transport steam, the monitoring operation ends without any message.

Parameter:
listener –Listener object to receive the event when an information change is detected.
bouquetId – Bouquet ID to monitor the information.
Exception:
SIIllegalArgumentException - In case the identifier is an invalid value as it is outside the scope or for any other reasons.
Reference:
SIMonitoringListener, SIMonitoringEvent

addBroadcasterMonitoringListener
public void **addBroadcasterMonitoringListener**(SIMonitoringListener listener,
int broadcasterId)
throws SIIllegalArgumentException

This method initializes the monitoring operation of broadcaster information. When

broadcaster information is changed, the registered listener object is notified of the event.

The monitoring operation depends on the mounting status. Continued monitoring is not necessarily required. The event of change is notified as soon as possible even if delayed compared with the actual change on the stream due to the resources distribution for the monitoring operation scheduled between tables. This standard does not specify the minimum requirements of the SI table monitoring operation. The mounted system should be optimum but depends on the status. However, when an application embedded with the receiver or the ARIB application, for example, detects the change at SI information acquisition from the stream, only the notification of the change to the listener is required.

When the network interface relevant to the SIDatabase object starts to select the station for another transport steam, the monitoring operation ends without any message.

Parameter:
listener - Listener object to receive the event when information change is detected.
broadcasterId - Broadcaster ID of the broadcaster to monitor the information.
Exception:
SIIllegalArgumentException - In case the identifier is an invalid value as it is outside the scope or for any other reasons.
Reference:
SIMonitoringListener, SIMonitoringEvent

addEventPresentFollowingMonitoringListener
public void **addEventPresentFollowingMonitoringListener**(SIMonitoringListener listener,
int originalNetworkId,
int transportStreamId,
int serviceId)
throws SIIllegalArgumentException

This method initializes the monitoring operation of EIT [Present/Following] information. When information is changed, the registered listener object is notified of the event.

The scope of monitoring is defined by the original network ID, the transport stream ID and the service. The listener is notified of the EIT [Present/Following] information change within the scope.

The monitoring operation depends on the mounting status. Continued monitoring is not necessarily required. The event of change is notified as soon as possible even if delayed compared with the actual change on the stream due to the resources distribution for the monitoring operation scheduled between tables. This standard does not specify the minimum requirements of the SI table monitoring operation. The mounted system should be optimum but depends on the status. However, when an application embedded with the receiver or the ARIB

application, for example, detects the change at SI information acquisition from the stream, only the notification of the change to the listener is required.

When the network interface relevant to the SIDatabase object starts to select the station for another transport steam, the monitoring operation ends without any message.

Parameter:
listener - Listener object to receive the event when information change is detected.
originalNetworkId – Original network ID to specify the monitoring scope.
transportStreamId – Transport stream ID to specify the monitoring scope.
serviceId – Service ID to specify the monitoring scope.
Exception:
SIIllegalArgumentException - In case the identifier is an invalid value as it is outside the scope or for any other reasons.
Reference:
SIMonitoringListener, SIMonitoringEvent

addEventScheduleMonitoringListener
public void **addEventScheduleMonitoringListener**(SIMonitoringListener listener,
int originalNetworkId,
int transportStreamId,
int serviceId,
java.util.Date startTime,
java.util.Date endTime)
throws SIIllegalArgumentException,
SIInvalidPeriodException

This method initializes the monitoring operation of EIT [Schedule] Information. When information is changed, the registered listener object is notified of the event.

The scope of monitoring is defined by the original network ID, the transport stream ID, the service and the start time/end time of the schedule span. The listener is notified of the EIT [Schedule] information change within the scope.

The monitoring operation depends on the mounting status. Continued monitoring is not necessarily required. The event of change is notified as soon as possible even if delayed compared with the actual change on the stream due to the resources distribution for the monitoring operation scheduled between tables. This standard does not specify the minimum requirements of the SI table monitoring operation. The mounted system should be optimum but depends on the status. However, when an application embedded with the receiver or the ARIB application, for example, detects the change at SI information acquisition from the stream, only the notification of the change to the listener is required.

When the network interface relevant to the SIDatabase object starts to select the station for another transport steam, the monitoring operation ends without any message.

Parameter:
listener - Listener object to receive the event when information change is detected.
originalNetworkId – Original network ID to specify the monitoring scope.
transportStreamId – Transport stream ID to specify the monitoring scope.
serviceId – Service ID to specify the monitoring scope.
startTime – Start time of the scheduled span.
endTime – End time of the scheduled span.
Exception:
SIIllegalArgumentException - In case the identifier is an invalid value as it is outside the scope or for any other reasons.
SIInvalidPeriodException - In case the start time is before the end time.
Reference:
SIMonitoringListener, SIMonitoringEvent

addNetworkMonitoringListener
public void **addNetworkMonitoringListener**(SIMonitoringListener listener,
int networkId)
throws SIIllegalArgumentException

This method initializes the monitoring operation of network information. When network information is changed, the registered listener object is notified of the event.

The monitoring operation depends on the mounting status. Continued monitoring is not necessarily required. The event of change is notified as soon as possible even if delayed compared with the actual change on the stream due to the resources distribution for the monitoring operation scheduled between tables. This standard does not specify the minimum requirements of the SI table monitoring operation. The mounted system should be optimum but depends on the status. However, when an application embedded with the receiver or the ARIB application, for example, detects the change at SI information acquisition from the stream, only the notification of the change to the listener is required.

When the network interface relevant to the SIDatabase object starts to select the station for another transport steam, the monitoring operation ends without any message.

Parameter:
listener - Listener object to receive the event when information change is detected.
networkId – Network ID of the network to monitor information.
Exception:
SIIllegalArgumentException - In case the identifier is an invalid value as it is outside the scope or any other reasons.

Reference:
SIMonitoringListener, SIMonitoringEvent

addPMTServiceMonitoringListener
public void **addPMTServiceMonitoringListener**(SIMonitoringListener listener,
int originalNetworkId,
int transportStreamId,
int serviceId)
throws SIIllegalArgumentException

This method initializes the monitoring operation of PMT information relevant to the service. When information is changed, the registered listener object is notified of the event. The monitoring operation depends on the mounting status. Continued monitoring is not necessarily required. The event of change is notified as soon as possible even if delayed compared with the actual change on the stream due to the resources distribution for the monitoring operation scheduled between tables. This standard does not specify the minimum requirements of the SI table monitoring operation. The mounted system should be optimum but depends on the status. However, when an application embedded with the receiver or the ARIB application, for example, detects the change at SI information acquisition from the stream, only the notification of the change to the listener is required.

When the network interface relevant to the SIDatabase object starts to select the station for another transport steam, the monitoring operation ends without any message.

Parameter:
listener - Listener object to receive the event when information change is detected.
originalNetworkId – Original network ID of the service
transportStreamId – Transport stream ID of the service
serviceId – Service ID of the service to monitor information
Exception:
SIIllegalArgumentException - In case the identifier is an invalid value as it is outside the scope or for any other reasons.
Reference:
SIMonitoringListener, SIMonitoringEvent

addServiceMonitoringListener
public void **addServiceMonitoringListener**(SIMonitoringListener listener,
int originalNetworkId,
int transportStreamId)
throws SIIllegalArgumentException

This method initializes the monitoring operation of SDT information relevant to the

service. When information relevant to the service is changed, the registered listener object is notified of the event.

The scope of monitoring is defined by the original network ID, the transport stream ID. The listener is notified of the information changes in various services within the scope.

The monitoring operation depends on the mounting status. Continued monitoring is not necessarily required. The event of change is notified as soon as possible even if delayed compared with the actual change on the stream due to the resources distribution for the monitoring operation scheduled between tables. This standard does not specify the minimum requirements of the SI table monitoring operation. The mounted system should be optimum but depends on the status. However, when an application embedded with the receiver or the ARIB application, for example, detects the change at SI information acquisition from the stream, only the notification of the change to the listener is required.

When the network interface relevant to the SIDatabase object starts to select the station for another transport steam, the monitoring operation ends without any message.

Parameter:
listener - Listener object to receive the event when information change is detected.
originalNetworkId – Original network ID to specify the monitoring scope
transportStreamId – Transport stream ID to specify the monitoring scope
Exception:
SIIllegalArgumentException - In case the identifier is an invalid value as it is outside the scope or for any other reasons.
Reference:
SIMonitoringListener, SIMonitoringEvent

getSIDatabase
public static SIDatabase[] **getSIDatabase**()

This method returns the array of the SIDatabase object (for each network interface). If there is only one network interface in the system, the number of elements for the array is "1". The network interface of each SIDatabase is used as data resources of all new data that are accessed by SIDatabase and/or SIInformation instance acquired from the interface. In case of transmission by PID of a different SI table, the SIDatabase instance uses the table data on the hierarchy that is currently selected by the network interface.

This method is the first one to access the ARIB-SI API and SIDatabase object to be returned and provides the access point to ARIB-SI information.

Return value:
Array of one SIDatabase object for each network interface

removeBouquetMonitoringListener
public void **removeBouquetMonitoringListener**(SIMonitoringListener listener,
int bouquetId)
throws SIIllegalArgumentException

This method removes the event listener registry for bouquet information monitoring. If this method is called without having the same SI object IDs for the argument as the one of the registered listener and the one of its registration time, the method fails and the listener remains registered.

Parameter:

listener – Listener object that has already been registered.

bouquetId – Bouquet ID to halt the information monitoring operation.

Exception :

SIIllegalArgumentException - In case the identifier is an invalid value as it is outside the scope or for any other reasons.

Reference:

SIMonitoringListener, SIMonitoringEvent

removeBroadcasterMonitoringListener
public void **removeBroadcasterMonitoringListener**(SIMonitoringListener listener,
int broadcasterId)
throws SIIllegalArgumentException

This method removes the event listener registry for broadcaster information monitoring. If this method is called without having the same SI object IDs for the argument as the one of the registered listener and the one of its registration time, the method fails and the listener remains registered.

Parameter:

listener - Listener object that has already been registered.

broadcasterId - Broadcaster ID to stop the information monitoring operation

Exception:

SIIllegalArgumentException - In case the identifier is an invalid value as it is outside the scope or for any other reasons.

Reference:

SIMonitoringListener, SIMonitoringEvent

removeEventPresentFollowingMonitoringListener
public void
removeEventPresentFollowingMonitoringListener(SIMonitoringListener listener,
int originalNetworkId,
int transportStreamId,
int serviceId)

throws SIIllegalArgumentException

This method removes the event listener registry for EIT[Present/Following] information monitoring. If this method is called without having the same SI object IDs for the argument as the one of the registered listener and the one of its registration time, the method fails and the listener remains registered.

Parameter:

listener - Listener object that has already been registered.
originalNetworkId – Original network ID to specify the monitoring scope.
transportStreamId – Transport stream ID to specify the monitoring scope.
serviceId – Service ID to specify the monitoring scope.

Exception:

SIIllegalArgumentException - In case the identifier is invalid as it is outside the scope or for any other reasons.

Reference:

SIMonitoringListener, SIMonitoringEvent

removeEventScheduleMonitoringListener

public void **removeEventScheduleMonitoringListener**(SIMonitoringListener listener,
int originalNetworkId,
int transportStreamId,
int serviceId)
throws SIIllegalArgumentException

Corresponding to the whole scheduled span, this method removes the event listener registry for EIT[Present/Following] information monitoring. If this method is called without having the same SI object IDs for the argument as the one of the registered listener and the one of its registration time, the method fails and the listener remains registered.

Parameter:

listener - Listener object that has already been registered.
originalNetworkId - Original network ID to specify the monitoring scope.
transportStreamId - Transport stream ID to specify the monitoring scope.
serviceId - Service ID to specify the monitoring scope.

Exception:

SIIllegalArgumentException - In case the identifier is invalid as it is outside the scope or for any other reasons.

Reference:

SIMonitoringListener, SIMonitoringEvent

removeNetworkMonitoringListener

public void **removeNetworkMonitoringListener**(SIMonitoringListener listener,

int networkId)
throws SIIllegalArgumentException

This method removes the event listener registry for network information monitoring. If this method is called without having the same SI object IDs for the argument as the one of the registered listener and the one of its registration time, the method fails and the listener remains registered.

Parameter:

listener - Listener object that has already been registered.
networkId – Network ID of the network to stop the monitoring operation.

Exception:

SIIllegalArgumentException - In case the identifier is invalid as it is outside the scope or for any other reasons.

Reference:

SIMonitoringListener, SIMonitoringEvent

removePMTServiceMonitoringListener
public void **removePMTServiceMonitoringListener**(SIMonitoringListener listener,
int originalNetworkId,
int transportStreamId,
int serviceId)
throws SIIllegalArgumentException

This method removes the event listener registry for PMT information monitoring relevant to the service. If this method is called without having the same SI object IDs for the argument as the one of the registered listener and the one of its registration time, the method fails and the listener remains registered.

Parameter:

listener - Listener object that has already been registered.
originalNetworkId – Original network ID of the service
transportStreamId – Transport stream ID of the service
serviceId – Service ID of the service for information monitoring

Exception:

SIIllegalArgumentException - In case the identifier is invalid as it is outside the scope or for any other reasons.

Reference:

SIMonitoringListener, SIMonitoringEvent

removeServiceMonitoringListener
public void **removeServiceMonitoringListener**(SIMonitoringListener listener,
int originalNetworkId,
int transportStreamId)

throws SIIllegalArgumentException

This method removes the event listener registry for information monitoring relevant to the service. If this method is called without having the same SI object IDs for the argument as the one of the registered listener and the one of its registration time, the method fails and the listener remains registered.

Parameter:

listener - Listener object that has already been registered.

originalNetworkId - Original network ID to specify the monitoring scope.

transportStreamId - Transport stream ID to specify the monitoring scope.

Exception:

SIIllegalArgumentException - In case the identifier is invalid as it is outside the scope or for any other reasons.

Reference:

SIMonitoringListener, SIMonitoringEvent

retrieveActualSINetwork

public SIRequest **retrieveActualSINetwork**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This method acquires information relevant to the present network. The present network transmits the network interface connected to the SIDatabase via the transport stream of the currently selected station.

In the SIIterator to be returned as the normal end event of acquisition requests, an object mounted with the SINetwork interface is included. When there is no corresponding object, either SIObjectNotInCacheEvent or SITableNotFoundEvent, whichever is suitable, is returned.

Parameter:

retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (always from the stream).

appData – Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, “null” is assigned.

listener - SIRetrievalListener to receive the event of request completion notification.

someDescriptorTags - Hint list of descriptors requested by the application (the descriptor is identified by the tag value). If there is one element of array and the value is RETRIEVE_ALL_INFORMATIONS, this means the application requests all

descriptors. If someDescriptorTags indicates “null,” this means the application does not need any of the descriptors. Any value that is not in the effective range of descriptor (0-255) is ignored, unless the array element is one and the value is RETRIEVE_ALL_INFORMATIONS

Return value:

SIRequest object

Exception:

SIIllegalArgumentException – In case the retrieveMode is an invalid value.

Reference:

SIRequest, SIRetrievalListener, SINetwork, DescriptorTag

retrieveActualSIServices

public SIRequest **retrieveActualSIServices**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This method acquires information relevant to the actual service. The actual service means the service being transmitted via the transport stream currently selected by the network interface connected to SIDatabase.

In SIIterator to be returned with the normal end event of acquisition requests, an object mounted with SIService interface is included. When there is no corresponding object, a suitable one among SIObjectNotInCacheEvent, SIObjectNotInTableEvent or SITableNotFoundEvent is returned.

Parameter:

retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (always from the stream).

appData – Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, “null” is assigned.

listener - SIRetrievalListener to receive the event of request completion notification.

someDescriptorTags - Hint list of descriptors requested by the application (the descriptor is identified by the tag value). In case there is one element of array and the value is RETRIEVE_ALL_INFORMATIONS, this means the application requests all descriptors. If someDescriptorTags indicates “null,” this means the application does not need any of the descriptors. Any value that is not in the effective range of descriptor (0-255) is ignored, unless the array element is one and the value is RETRIEVE_ALL_INFORMATIONS

Return value:
SIRequest object
Exception:
SIIllegalArgumentException – In case the retrieveMode is an invalid value.
Reference:
SIRequest, SIRetrievalListener, SIService, DescriptorTag

retrieveActualSITransportStream
public SIRequest **retrieveActualSITransportStream**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This method acquires information relevant to the actual transport stream. Actual transport stream means the transport stream to which the network interface connected to the SIDatabase is selecting.

In SIIterator to be returned as the normal end event of acquisition requests, an object mounted with SITransportStreamNIT interface is included. When there is no corresponding object, a suitable one among SIObjectNotInCacheEvent, SIObjectNotInTableEvent or SITableNotFoundEvent is returned.

Parameter:
retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (always from the stream).
appData – Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, “null” is assigned.
listener - SIRetrievalListener to receive the event of request completion notification.
someDescriptorTags - Hint list of descriptors requested by the application (the descriptor is identified by the tag value). In case there is one element of array and the value is RETRIEVE_ALL_INFORMATIONS, this means the application requests all descriptors. If someDescriptorTags indicates “null,” this means the application does not need any of the descriptors. Any value that is not in the effective range of descriptor (0-255) is ignored, unless the array element is one and the value is RETRIEVE_ALL_INFORMATIONS
Return value:
SIRequest object
Exception:
SIIllegalArgumentException – In case the retrieveMode is an invalid value.

Reference:
SIRequest, SIRetrievalListener, SITransportStream, DescriptorTag

retrievePMTElementaryStreams
public SIRequest **retrievePMTElementaryStreams**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
ARIBLocator aribLocator,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This method acquires PMT elementary stream information relevant to the service component from the actual transport stream of this SIDatabase object. The required component can be designated by the ARIBLocator.

In SIIterator to be returned as the normal end event of acquisition requests, an object mounted with the PMTElementaryStream interface is included. When there is no corresponding object, a suitable one among ObjectNotInCacheEvent, ObjectNotInTableEvent or TableNotFoundEvent SIObjectNotInCacheEvent is returned.

Parameter:
retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (always from the stream).
appData –Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, “null” is assigned.
listener - SIRetrievalListener to receive the event of request completion notification.
aribLocator – ARIBLocator to identify the component of service. This locator may include more information than the service component identification. This method uses only the section from the top to the component tag.
someDescriptorTags - Hint list of descriptors requested by the application (the descriptor is identified by the tag value). In case there is one element of array and the value is RETRIEVE_ALL_INFORMATIONS, this means the application requests all descriptors. If someDescriptorTags indicates “null,” this means the application does not need any of the descriptors. Any value that is not in the effective range of descriptor (0-255) is ignored, unless the array element is one and the value is RETRIEVE_ALL_INFORMATIONS
Return value:
SIRequest object
Exception:

SIIllegalArgumentException – In case the retrieveMode is an invalid value, the locator is invalid or does not identify the service component.

Reference:

SIRequest, SIRetrievalListener, SIService, DescriptorTag

retrievePMTElementaryStreams

public SIRequest **retrievePMTElementaryStreams**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
int serviceId,
int componentTag,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This method acquires PMT elementary stream information relevant to the service component from the actual transport stream of this SIDatabase object. The elementary stream to be requested can be designated by its ID. If RETRIEVE_ALL_INFORMATIONS is designated for the argument componentTag, all elementary streams, regardless of the tag value, are acquired. If RETRIEVE_CURRENT_SELECTED is designated, the elementary stream that is currently selected is acquired.

In SIIterator to be returned as the normal end event of acquisition requests, an object mounted with the PMTElementaryStream interface is included. When there is no corresponding object, a suitable one among SIObjectNotInCacheEvent, SIObjectNotInTableEvent or SITableNotFoundEvent is returned.

Parameter:

retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (always from the stream).

appData – Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, “null” is assigned.

listener - SIRetrievalListener to receive the event of request completion notification.

serviceId – Service id to identify the elementary stream to be acquired.

componentTag – Component tag to identify the elementary stream to be acquired. In case of RETRIEVE_ALL_INFORMATIONS, all elementary streams are returned and in case of RETRIEVE_CURRENT_SELECTED, the elementary stream that is currently selected is returned.

someDescriptorTags - Hint list of descriptors requested by the application (the descriptor is identified by the tag value). If there is one element of array and the value is

RETRIEVE_ALL_INFORMATIONS, this means the application requests all descriptors. If someDescriptorTags indicates “null,” this means the application does not need any of the descriptors. Any value that is not in the effective range of descriptor (0-255) is ignored, unless the array element is one and the value is RETRIEVE_ALL_INFORMATIONS

Return value:

SIRequest object

Exception:

SIIllegalArgumentException – In case the retrieveMode is an invalid value or each identification value is outside the scope.

Reference:

SIRequest, SIRetrievalListener, SIService, DescriptorTag

retrievePMTService

public SIRequest **retrievePMTService**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
ARIBLocator aribLocator,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This method acquires PMT information relevant to the service. The requested service can be designated by the ARIBLocator.

In SIIterator to be returned as the normal end event of acquisition requests, an object mounted with PMTService interface is included. When there is no corresponding object, a suitable one among SIObjectNotInCacheEvent, SIObjectNotInTableEvent or SITableNotFoundEvent is returned.

Parameter:

retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (always from the stream).

appData – Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, “null” is assigned.

listener - SIRetrievalListener to receive the event of request completion notification.

aribLocator - ARIBLocator to identify the service. This locator may include more information than the service identification. This method uses only the section from the top to the service ID.

someDescriptorTags - Hint list of descriptors requested by the application (the descriptor is identified by the tag value). If there is one element of array and the value is

RETRIEVE_ALL_INFORMATIONS, this means the application requests all descriptors. If someDescriptorTags indicates “null,” this means the application does not need any of the descriptors. Any value that is not in the effective range of descriptor (0-255) is ignored, unless the array element is one and the value is RETRIEVE_ALL_INFORMATIONS

Return value:

SIRequest object

Exception:

SIIllegalArgumentException - In case the retrieveMode is an invalid value, or the locator is invalid or does not identify the service.

Reference:

SIRequest, SIRetrievalListener, SIService, DescriptorTag

retrievePMTServices

public SIRequest **retrievePMTServices**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
int serviceId,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This method acquires PMT information relevant to the service from the actual transport stream of this SIDatabase object. The service to be requested can be designated by its ID. If RETRIEVE_ALL_INFORMATIONS is designated for the argument serviceId, all services, regardless of the service ID, are acquired. If RETRIEVE_CURRENT_SELECTED is designated, the service that is currently selected is acquired.

In SIIterator to be returned with the normal end event of acquisition requests, an object mounted with PMTService interface is included. When there is no corresponding object, a suitable one among SIObjectNotInCacheEvent, SIObjectNotInTableEvent or SITableNotFoundEvent is returned.

Parameter:

retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (always from the stream).

appData – Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, “null” is assigned.

listener - SIRetrievalListener to receive the event of request completion notification.

serviceId – Service ID to identify the service to be acquired. In case of

RETRIEVE_ALL_INFORMATIONS, all services, regardless of the service ID, are returned. In case of RETRIEVE_CURRENT_SELECTED, the service of the station currently selected is returned.

someDescriptorTags - Hint list of descriptors requested by the application (the descriptor is identified by the tag value). If there is one element of array and the value is RETRIEVE_ALL_INFORMATIONS, this means the application requests all descriptors. If someDescriptorTags indicates "null," this means the application does not need any of the descriptors. Any value that is not in the effective range of descriptor (0-255) is ignored, unless the array element is one and the value is RETRIEVE_ALL_INFORMATIONS

Return value:

SIRequest object

Exception:

SIIllegalArgumentException – In case the retrieveMode is an invalid value or the value of service ID is outside the scope.

Reference:

SIRequest, SIRetrievalListener, SIService, DescriptorTag

retrieveSIBouquets

public SIRequest **retrieveSIBouquets**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
int bouquetId,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This method acquires information relevant to the bouquet. The bouquet can be designated by its ID. If RETRIEVE_ALL_INFORMATIONS is designated for the argument bouquetId, all bouquets in BAT that the transport stream on the network interface is currently receiving are acquired.

In SIIterator to be returned as the normal end event of acquisition requests, an object mounted with SIBouquet interface is included. When there is no corresponding object, a suitable one among SIObjectNotInCacheEvent, SIObjectNotInTableEvent or SITableNotFoundEvent is returned.

Parameter:

retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (always from the stream).

appData – Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, "null" is

assigned.
listener - SIRetrievalListener to receive the event of request completion notification.
bouquetId – Either bouquet Id to identify the bouquet to be acquired or RETRIEVE_ALL_INFORMATIONS to acquire all bouquets on the current transport stream.
someDescriptorTags - Hint list of descriptors requested by the application (the descriptor is identified by the tag value). In case there is one element of array and the value is RETRIEVE_ALL_INFORMATIONS, this means the application requires all descriptors. If someDescriptorTags indicates "null," this means the application does not need any of the descriptors. Any value that is not in the effective range of descriptor (0-255) is ignored, unless the array element is one and the value is RETRIEVE_ALL_INFORMATIONS

Return value:

SIRequest object

Exception:

SIIllegalArgumentException - In case the retrieveMode is an invalid value or the value of service ID is outside the scope.

Reference:

SIRequest, SIRetrievalListener, SIBouquet, DescriptorTag

retrieveSIBroadcaster

public SIRequest **retrieveSIBroadcaster**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
int originalNetworkId,
int broadcasterId,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This method acquires information relevant to the broadcaster. The broadcaster can be designated by its ID. If RETRIEVE_ALL_INFORMATIONS is designated for the argument broadcasterId, all broadcasters in BAT that the transport stream on the network interface is currently receiving are acquired.

In SIIterator to be returned as the normal end event of the requests, one or more object mounted with SIBroadcaster interface is included. When there is no corresponding object, a suitable one among SIObjectNotInCacheEvent, SIObjectNotInTableEvent or SITableNotFoundEvent is returned.

Parameter:

retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (always from the stream).

appData – Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, “null” is assigned.
listener - SIRetrievalListener to receive the event of request completion notification.
originalNetworkId – Original network ID that includes the broadcaster to be acquired.
broadcasterId – Either broadcaster ID to identify the bouquet to be acquired or RETRIEVE_ALL_INFORMATIONS to acquire all broadcaster on the current transport stream.
someDescriptorTags - Hint list of descriptors requested by the application (the descriptor is identified by the tag value). In case there is one element of array and the value is RETRIEVE_ALL_INFORMATIONS, this means the application requires all descriptors. If someDescriptorTags indicates “null,” this means the application does not need any of the descriptors. Any value that is not in the effective range of descriptor (0-255) is ignored, unless the array element is one and the value is RETRIEVE_ALL_INFORMATIONS

Return value:
SIRequest object

Exception:
SIIllegalArgumentException – In case the retrieveMode is an invalid value or the broadcaster ID is outside the scope.

Reference:
SIRequest, SIRetrievalListener, SIBroadcaster, DescriptorTag

retrieveSIBroadcasters
public SIRequest **retrieveSIBroadcasters**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
int originalNetworkId,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This method acquires information relevant to the broadcaster specified by originalNetworkId.

In SIIterator to be returned with the normal end event of acquisition requests, an object mounted with SIBroadcaster interface is included. When there is no corresponding object, a suitable one among SIObjectNotInCacheEvent, SIObjectNotInTableEvent or SITableNotFoundEvent is returned.

Parameter:
retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If

not, from the stream) or FROM_STREAM_ONLY (always from the stream).

appData – Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, "null" is assigned.

listener - SIRetrievalListener to receive the event of request completion notification.
originalNetworkId – Original network ID comprised of acquired broadcasters.

someDescriptorTags - Hint list of descriptors requested by the application (the descriptor is identified by the tag value). If there is one element of array and the value is RETRIEVE_ALL_INFORMATIONS, this means the application requests all descriptors. If someDescriptorTags indicates "null," this means the application does not need any of the descriptors. Any value that is not in the effective range of descriptor (0-255) is ignored, unless the array element is one and the value is RETRIEVE_ALL_INFORMATIONS

Return value:
SIRequest object

Exception:
SIIllegalArgumentException - In case the value of the retrieveMode is an invalid value or the value of the original network ID is outside the scope.

Reference:
SIRequest, SIRetrievalListener, SIBroadcaster, DescriptorTag

retrieveSINetworks
public SIRequest **retrieveSINetworks**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
int networkId,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This method acquires information relevant to the network. The network can be designated by its ID. If RETRIEVE_ALL_INFORMATIONS is designated for the argument networkId, all networks in NIT that actual TS and non-actual TS on the network interface are currently receiving are acquired. If RETRIEVE_CURRENT_SELECTED is designated, the network that is currently selected is acquired.

In SIIterator to be returned as the normal end event of acquisition requests, an object mounted with more that one SINetwork interfaces is included. When there is no corresponding object, either SIObjectNotInCacheEvent or SITableNotFoundEvent, whichever is suitable, is returned.

Parameter:

retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (always from the stream).

appData – Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, "null" is assigned.

listener - SIRetrievalListener to receive the event of request completion notification.

networkId – Network ID of the network to be acquired. If the value is RETRIEVE_ALL_INFORMATIONS, all networks that are currently transmitted are returned. If RETRIEVE_CURRENT_SELECTED, the network currently selected is returned.

someDescriptorTags - Hint list of descriptors requested by the application (the descriptor is identified by the tag value). If there is one element of array and the value is RETRIEVE_ALL_INFORMATIONS, this means the application requests all descriptors. If someDescriptorTags indicates "null," this means the application does not need any of the descriptors. Any value that is not in the effective range of descriptor (0-255) is ignored, unless the array element is one and the value is RETRIEVE_ALL_INFORMATIONS

Return value:

SIRequest object

Exception:

SIIllegalArgumentException - In case the value retrieveMode is an invalid value or the network ID is outside the scope.

Reference:

SIRequest, SIRetrievalListener, SINetwork, DescriptorTag

retrieveSIService

public SIRequest **retrieveSIService**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
ARIBLocator aribLocator,

short[] someDescriptorTags)
throws SIIllegalArgumentException

This method acquires information relevant to the service. The required service can be designated by the ARIBLocator.

In SIIterator to be returned as the normal end event of acquisition requests, an object mounted with SIService interface is included. When there is no corresponding object, a suitable one among SIObjectNotInCacheEvent, SIObjectNotInTableEvent or SITableNotFoundEvent is returned.

Parameter:
retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (always from the stream).

appData – Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, “null” is assigned.

listener - SIRetrievalListener to receive the event of request completion notification.
aribLocator – ARIBLocator to identify the component of service. This locator may include more information than the service component identification. This method uses only the section from the top to the component tag.

someDescriptorTags - Hint list of descriptors requested by the application (the descriptor is identified by the tag value). If there is one element of array and the value is RETRIEVE_ALL_INFORMATIONS, this means the application requests all descriptors. If someDescriptorTags indicates “null,” this means the application does not need any of the descriptors. Any value that is not in the effective range of descriptor (0-255) is ignored, unless the array element is one and the value is RETRIEVE_ALL_INFORMATIONS.

Return value:
SIRequest object
Exception:
SIIllegalArgumentException - In case the value of the retrieveMode is an invalid value or each identification value for service identification is outside the scope.
Reference:
SIRequest, SIRetrievalListener, SIService, DescriptorTag

retrieveSIServices
public SIRequest **retrieveSIServices**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
int originalNetworkId,
int transportStreamId,
int serviceId,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This method acquires information relevant to the service. The service can be designated by its ID. If RETRIEVE_ALL_INFORMATIONS is designated for the argument transportStreamId, all elementary streams, regardless of the value of transport stream ID, are acquired. If RETRIEVE_CURRENT_SELECTED is designated, the service on the transport steam that is currently selected is acquired. If RETRIEVE_ALL_INFORMATIONS is designated for the argument serviceId, all services, regardless of the service ID value, are acquired. If RETRIEVE_CURRENT_SELECTED is designated, the service of which the station is currently selected is returned.

In SIIterator to be returned with the normal end event of acquisition requests, an object mounted with more that one SIService interfaces is included. When there is no corresponding object, a suitable one among SIObjectNotInCacheEvent, SIObjectNotInTableEvent or SITableNotFoundEvent is returned.

Parameter:

retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (always from the stream).

appData – Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, “null” is assigned.

listener - SIRetrievalListener to receive the event of request completion notification.
originalNetworkId – Original network ID to identify the service to be acquired.
transportStreamId – Transport stream ID to identify the service to be acquired. In case of RETRIEVE_ALL_INFORMATIONS, all services regardless of the transport stream ID are returned. In case of RETRIEVE_CURRENT_SELECTED, the service on the transport stream that is currently selected.

serviceId – Service ID to identify the service to be acquired. In case of

RETRIEVE_ALL_INFORMATIONS, all services regardless of the transport stream ID are returned. In case of RETRIEVE_CURRENT_SELECTED, the service on the transport stream that is currently selected.

someDescriptorTags - Hint list of descriptors required by the application (the descriptor is identified by the tag value). In case there is one element of array and the value is RETRIEVE_ALL_INFORMATIONS, this means the application requests all descriptors. If someDescriptorTags indicates "null," this means the application does not need any of the descriptors. Any value that is not in the effective range of descriptor (0-255) is ignored, unless the array element is one and the value is RETRIEVE_ALL_INFORMATIONS.

Return value:
SIRequest object
Exception:
SIIllegalArgumentException - In case the retrieveMode is an invalid value or each ID value is outside the scope.
Reference:
SIRequest, SIRetrievalListener, SIService, DescriptorTag

retrieveSITimeFromTDT
public SIRequest **retrieveSITimeFromTDT**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener)
throws SIIllegalArgumentException

This method acquires information relevant to the time from Time Date Table (TDT) on the actual transport stream.

In SITime to be returned as the normal end event of acquisition requests, an object mounted with the PMTElementaryStream interface is included. When there is no corresponding object, a suitable one among SIObjectNotInCacheEvent, SIObjectNotInTableEvent and SITableNotFoundEvent is returned.

Parameter:
retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (always from the stream).
appData – Optional object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, "null" is assigned.
listener - SIRetrievalListener to receive the event of request completion notification.

Return value:
SIRequest object
Exception:
SIIllegalArgumentException – In case the retrieveMode is an invalid value.
Reference:
SIRequest, SIRetrievalListener, SITime

retrieveSITimeFromTOT
public SIRequest **retrieveSITimeFromTOT**(short retrieveMode,
java.lang.Object appData,
SIRetrievalListener listener,
short[] someDescriptorTags)
throws SIIllegalArgumentException

This method acquires information relevant to the time from Time Offset Table (TOT) on the actual transport stream. Offset information follows the time information.

In SITime to be returned as the normal end event of acquisition requests, an object mounted with the PMTElementaryStream interface is included. When there is no corresponding object, a suitable one among SIObjectNotInCacheEvent, SIObjectNotInTableEvent and SITableNotFoundEvent is returned.

Parameter:
retrieveMode - specifies the data acquisition mode. Any of FROM_CACHE_ONLY (only from the cache), FROM_CACHE_OR_STREAM (from the cache if it is usable. If not, from the stream) or FROM_STREAM_ONLY (only from the stream).

appData – Object to be provided from the application. It is transferred to the listener when the request is completed. The application can use this object for inter-communication. If the application does not need this kind of data, “null” is assigned.

listener - SIRetrievalListener to receive the event of request completion notification.

someDescriptorTags - Hint list of descriptors requested by the application (the descriptor is identified by the tag value). In case there is one element of array and the value is RETRIEVE_ALL_INFORMATIONS, this means the application requests all descriptors. If someDescriptorTags indicates “null,” this means the application does not need any of the descriptors. Any value that is not in the effective range of descriptor (0-255) is ignored, unless the array element is one and the value is RETRIEVE_ALL_INFORMATIONS.
Return value:
SIRequest object

Exception:
SIIllegalArgumentException – In case the retrieveMode is an invalid value.
Reference:
SIRequest, SIRetrievalListener, SITime

M.2.24 SIExEventInformation
jp.or.arib.tv.si
Class SIExEventInformation

java.lang.Object
|
+--**jp.or.arib.tv.si.SIExEventInformation**

public class **SIExEventInformation**
extends java.lang.Object

This interface indicates the description items and name of details of a specific program. The information is acquired from extended format event descriptors.

General information on the method	
java.lang.String	**getDescription**() This method acquires the item description.
java.lang.String	**getName**() This method acquires the item name.

Method inherited from Class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Details of the method

getDescription
public java.lang.String **getDescription**()
This method acquires the item description.
Return value:
The item description for the extended information of the program

getName
public java.lang.String **getName**()
This method acquires the name of item.
Return value:

Item name corresponding to the extended information of program

M.2.25 SILackOfResourcesEvent
jp.or.arib.tv.si
Class SILackOfResourcesEvent

java.lang.Object
  |
  +--java.util.EventObject
  |
  +--jp.or.arib.tv.si.SIRetrievalEvent
  |
  +--**jp.or.arib.tv.si.SILackOfResourcesEvent**

Mounted interface for all:
java.io.Serializable

public class **SILackOfResourcesEvent**
extends SIRetrievalEvent

This event is notified in case necessary resources to acquire the data are not available at SI acquisition request. For instance, in case necessary resources are occupied by the called application itself or all other applications.

Reference:
SIRetrievalListener, java.io.Serializable

General information on the constructor
SILackOfResourcesEvent(java.lang.Object appData, SIRequest request) The constructor of this event

Method inherited from Class jp.or.arib.tv.si.SIRetrievalEvent
getAppData, getSource

Method inherited from Class java.util.EventObject
toString

Method inherited from Class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Details of the constructor

SILackOfResourcesEvent

public **SILackOfResourcesEvent**(java.lang.Object appData,
SIRequest request)

The constructor of this event.

Parameter:

appData – Application data given to the request method

request – Instance of the event notifying SIRequest

M.2.26 SIMonitoringEvent

jp.or.arib.tv.si

Class SIMonitoringEvent

java.lang.Object
|
+--java.util.EventObject
|
+--**jp.or.arib.tv.si.SIMonitoringEvent**

Mounted interface for all:

java.io.Serializable

public class **SIMonitoringEvent**
extends java.util.EventObject

The object of this class is transmitted for the listener object to be used to notify the application of the monitored information change.

Reference:

SIMonitoringType, SIMonitoringListener, java.io.Serializable

General information on the constructor

SIMonitoringEvent(SIDatabase source, byte objectType, int networkId, int bouquetId, int originalNetworkId, int transportStreamId, int broadcasterId, int serviceId, java.util.Date startTime, java.util.Date endTime) Constructor of event object

General information on the method	
int	**getBouquetID**() This method returns the bouquet ID.
int	**getBroadcasterID**() This method returns the broadcaster ID of the broadcaster.
java.util.Date	**getEndTime**() This method returns the end time of the schedule span when the event information in the schedule span is changed.
int	**getNetworkID**() This method returns the network ID of the network.
int	**getOriginalNetworkID**() This method returns the original network ID of the SIInformation object.
int	**getServiceID**() This method returns the service ID of the SIInformation object.
byte	**getSIInformationType**() This method acquires the type of the SIInformation at the information change.
java.lang.Object	**getSource**() This method acquires SIDatabase instance to be sent to the event.
java.util.Date	**getStartTime**() This method returns the start time of the schedule span when the event information in the schedule span is changed.
int	**getTransportStreamID**() This method returns the transport stream ID of the SIInformation object.

Method inherited from Class java.util.EventObject
toString

Method inherited from Class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Details of the constructor

SIMonitoringEvent

public **SIMonitoringEvent**(SIDatabase source,
byte objectType,
int networkId,
int bouquetId,
int originalNetworkId,
int transportStreamId,
int broadcasterId,
int serviceId,

java.util.Date startTime,
java.util.Date endTime)

Constructor of the event object

Parameter:

source – Object to be an origin of event.
objectType - (including SIMonitoringType) Type of the SIInformation object
networkId - Network ID
bouquetId – Bouquet ID
originalNetworkId – Original network ID
transportStreamId – Transport stream ID
broadcasterId – Broadcaster ID
serviceId – Service ID
startTime – Start time of event schedule span
endTime – End time of event schedule span

Details of the method

getBouquetID
public int **getBouquetID**()

This method returns the bouquet ID of the bouquet. This method is applicable only when the SIInformation returned by getSIInformationType is BOUQUET.

Return value:

Bouquet ID. In case it is not applicable to this event, the value is "-2".

getBroadcasterID
public int **getBroadcasterID**()

This method returns the broadcaster ID of the broadcaster. This method is applicable only when the SIInformation returned by getSIInformationType is BROADCASTER.

Return value:

Broadcaster ID. In case it is not applicable to this event, the value is "-2".

getEndTime
public java.util.Date **getEndTime**()

This method returns the end time of the schedule span when the event information in the schedule span is changed. This method is applicable only when the SIInformation returned by getSIInformationType is SCHEDULED_EVENT.

Return value:

End time. In case it is not applicable to this event, the value is "null".

getNetworkID
public int **getNetworkID**()

This method returns the network ID of the network. This method is applicable only when the SIInformation returned by getSIInformationType is NETWORK.

Return value:
Network ID. In case it is not applicable to this event, the value is "-2".

getOriginalNetworkID
public int **getOriginalNetworkID**()

This method returns the original network ID of the SIInformation object. This method is applicable only when the SIInformation returned by getSIInformationType is SERVICE, PMT_SERVICE, PRESENT_FOLLOWING_EVENT or SCHEDULED_EVENT.

Return value:
Original network ID. In case it is not applicable to this event, the value is "-2".

getServiceID
public int **getServiceID**()

This method returns the service ID of the SIInformation object. This method is applicable only when the SIInformation returned by getSIInformationType is SERVICE, PMT_SERVICE, PRESENT_FOLLOWING_EVENT or SCHEDULED_EVENT.

Return value:
Service ID. In case it is not applicable to this event, the value is "-2".

getSIInformationType
public byte **getSIInformationType**()

This method acquires the type of the SIInformation at the information change.

Return value:
SIInformation type. SIMonitoringType interface defines the acquirable value.

References:
SIMonitoringType

getSource
public java.lang.Object **getSource**()

This method acquires SIDatabase instance to be sent to the event.

Override:
getSource in Class java.util.EventObject

Return value:
SIDatabase instance to be the origin of the event

getStartTime
public java.util.Date **getStartTime**()

This method returns the start time of the schedule span when the event information in the schedule span is changed. This method is applicable only when the SIInformation returned by getSIInformationType is CHEDULED_EVENT.

Return value:

Start time. In case it is not applicable to this event, the value is “null”.

getTransportStreamID
public int **getTransportStreamID**()

This method returns the transport stream ID of the SIInformation object. This method is applicable only when the SIInformation returned by getSIInformationType is SERVICE, PMT_SERVICE, PRESENT_FOLLOWING_EVENT or SCHEDULED_EVENT.

Return value:

Transport stream ID. In case it is not applicable to this event, the value is “-2”.

M.2.27 SINotInCacheEvent
jp.or.arib.tv.si
Class SINotInCacheEvent

java.lang.Object
  |
  +--java.util.EventObject
  |
  +--jp.or.arib.tv.si.SIRetrievalEvent
  |
  +--**jp.or.arib.tv.si.SINotInCacheEvent**

Mounted interface for all:

java.io.Serializable

public class **SINotInCacheEvent**
extends SIRetrievalEvent

When the SI acquisition request in FROM_CACHE_ONLY mode is performed and the requested data do not exist in the cache, this event is notified as a response.

Reference:

SIRetrievalListener, java.io.Serializable

General information on the constructor
SINotInCacheEvent(java.lang.Object appData, SIRequest request) Event constructor

Method inherited from Class jp.or.arib.tv.si.SIRetrievalEvent
getAppData, getSource

Method inherited from Class java.util.EventObject
toString

Method inherited from Class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Details of the constructor

SINotInCacheEvent
public **SINotInCacheEvent**(java.lang.Object appData,
SIRequest request)
Event constructor
Parameter:
appData – Application data given when the request method is called.
request – SIRequest of the event origin

M.2.28 SIObjectNotInTableEvent
jp.or.arib.tv.si
Class SIObjectNotInTableEvent

java.lang.Object
|
+--java.util.EventObject
|
+--jp.or.arib.tv.si.SIRetrievalEvent
|
+--**jp.or.arib.tv.si.SIObjectNotInTableEvent**

Mounted interface for all:
java.io.Serializable

public class **SIObjectNotInTableEvent**

extends SIRetrievalEvent

This event is notified when the SI table that has information about the location of the requested object is acquired for the SI acquisition request but the object is absent. The reason could be that the object corresponding to the requested ID is absent.

Reference:

SIRetrievalListener, java.io.Serializable

General information on the constructor
SIObjectNotInTableEvent(java.lang.Object appData, SIRequest request) Event constructor

Method inherited from Class jp.or.arib.tv.si.SIRetrievalEvent
getAppData, getSource

Method inherited from Class java.util.EventObject
toString

Method inherited from Class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Details of the constructor

SIObjectNotInTableEvent

public **SIObjectNotInTableEvent**(java.lang.Object appData,
SIRequest request)

Event constructor

Parameter:

appData - Application data given when the request method is called.

request – SIRequest instance of event origin

M.2.29 SIRequest

jp.or.arib.tv.si

Class SIRequest

java.lang.Object
|
+--**jp.or.arib.tv.si.SIRequest**

public class **SIRequest**
extends java.lang.Object

The instance object of this class indicates acquisition request from the application. The application may cancel the request using this object.

General information on the method	
boolean	**cancelRequest**() This method cancels the acquisition request.
boolean	**isAvailableInCache**() This method returns the availability of the information whether it is returned from the stream or cache.

Method inherited from Class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Details of the method

cancelRequest
public boolean **cancelRequest**()

This method cancels the acquisition request.

Return value:

If the request is canceled and the listener is notified of the SIRequestCancelEvent, the value is true. If the request has already been completed, the value is false (including any cases of normal end, abnormal end with error or cancel method calling of higher priority.)

isAvailableInCache
public boolean **isAvailableInCache**()

This method returns the availability of the stream whether it is returned from the stream or cache.

Return value:

If the information is from the cache, the value is “true”. If the information is from the stream, the value is “false”.

M.2.30 SIRequestCancelledEvent
jp.or.arib.tv.si
Class SIRequestCancelledEvent

java.lang.Object
  |
  +--java.util.EventObject
  |
  +--jp.or.arib.tv.si.SIRetrievalEvent
  |
  +--**jp.or.arib.tv.si.SIRequestCancelledEvent**

Mounted interface for all:
java.io.Serializable

public class **SIRequestCancelledEvent**
extends SIRetrievalEvent

This event is sent as a response when the request itself is cancelled by calling SIRequest.cancelRequest method at SI acquisition request.

Reference:
SIRequest, SIRetrievalListener, java.io.Serializable

General information on the constructor
SIRequestCancelledEvent(java.lang.Object appData, SIRequest request) Constructor

Method inherited from Class jp.or.arib.tv.si.SIRetrievalEvent
getAppData, getSource

Method inherited from Class java.util.EventObject
toString

Method inherited from Class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Details of the constructor

SIRequestCancelledEvent
public **SIRequestCancelledEvent**(java.lang.Object appData,
  SIRequest request)
Constructor

Parameter:

appData - Application data given when the request method is called.
request – SIRequest instance that is the origin of the event

M.2.31 SIRetrievalEvent
jp.or.arib.tv.si
Class SIRetrievalEvent

java.lang.Object
|
+--java.util.EventObject
|
+--**jp.or.arib.tv.si.SIRetrievalEvent**

Mounted interface for all:

java.io.Serializable

Known subclass of direct line:

SILackOfResourcesEvent, SINotInCacheEvent, SIObjectNotInTableEvent, SIRequestCancelledEvent, SISuccessfulRetrieveEvent, SITableNotFoundEvent, SITableUpdatedEvent

public abstract class **SIRetrievalEvent**
extends java.util.EventObject

Basic class for the event of the SI acquisition request completion. Only one event is returned for one SI acquisition request.

Reference:

SIRetrievalListener, java.io.Serializable

General information on the constructor
SIRetrievalEvent(java.lang.Object appData, SIRequest request) Event constructor

General information on the method	
java.lang.Object	**getAppData**() This method returns the application data given to the request method.
java.lang.Object	**getSource**()

	This method returns the event origin SIRequest.

Method inherited from Class java.util.EventObject
toString

Method inherited from Class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Details of the constructor

SIRetrievalEvent

public **SIRetrievalEvent**(java.lang.Object appData,
SIRequest request)

Event constructor

Parameter:

appData - Application data given when the request method is called.

request – SIRequest instance of event origin

Details of the method

getAppData

public java.lang.Object **getAppData**()

This method returns the application data given to the request method.

Return value:

Application data

getSource

public java.lang.Object **getSource**()

This method returns SIRequest of event origin.

Override:

getSource in Class java.util.EventObject

Return value:

SIRequest object

M.2.32 SISuccessfulRetrieveEvent

jp.or.arib.tv.si

Class SISuccessfulRetrieveEvent

java.lang.Object
|

+--java.util.EventObject
|
+--jp.or.arib.tv.si.SIRetrievalEvent
|
+--**jp.or.arib.tv.si.SISuccessfulRetrieveEvent**

Mounted interface for all:
java.io.Serializable

public class **SISuccessfulRetrieveEvent**
extends SIRetrievalEvent

This event is sent as a response when the request is normally completed for the SI acquisition request. The requested result can be acquired using getResult method.

Reference:
SIRetrievalListener, java.io.Serializable

General information on the constructor
SISuccessfulRetrieveEvent(java.lang.Object appData, SIRequest request, SIIterator result) Constructor

General information on the method	
SIIterator	**getResult**() This method returns SIIterator object that includes the requested data.

Method inherited from Class jp.or.arib.tv.si.SIRetrievalEvent
getAppData, getSource

Method inherited from Class java.util.EventObject
toString

Method inherited from Class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Details of the constructor

SISuccessfulRetrieveEvent
public **SISuccessfulRetrieveEvent**(java.lang.Object appData,

SIRequest request,
SIIterator result)

Constructor

Parameter:

appData – Application data given when the acquisition request method is called.
request – SIRequest instance that is the event origin
result – SIIterator that includes the acquired object

Details of the method

getResult
public SIIterator **getResult**()

This method returns SIIterator object that includes the requested data.

Return value:

SIIterator that includes the requested object

Reference:

SIObjectNotInTableEvent

M.2.33 SITableNotFoundEvent
jp.or.arib.tv.si
Class SITableNotFoundEvent

java.lang.Object
|
+--java.util.EventObject
|
+--jp.or.arib.tv.si.SIRetrievalEvent
|
+--**jp.or.arib.tv.si.SITableNotFoundEvent**

Mounted interface for all:

java.io.Serializable

public class **SITableNotFoundEvent**
extends SIRetrievalEvent

This event is sent as a response when the SI table that must include the requested information. The reason could be that the requested table is not broadcasted on the transport stream that is currently connected to SI database.

Reference:

SIRetrievalListener, java.io.Serializable

General information on the constructor
SITableNotFoundEvent(java.lang.Object appData, SIRequest request) Constructor

Method inherited from Class jp.or.arib.tv.si.SIRetrievalEvent
getAppData, getSource

Method inherited from Class java.util.EventObject
toString

Method inherited from Class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Details of the constructor

SITableNotFoundEvent

public **SITableNotFoundEvent**(java.lang.Object appData,
SIRequest request)

Constructor

Parameter:

appData - Application data given when the request method is called.

request – SIRequest instance that is the origin of the event

M.2.34 SITableUpdatedEvent

jp.or.arib.tv.si

Class SITableUpdatedEvent

java.lang.Object
|
+--java.util.EventObject
|
+--jp.or.arib.tv.si.SIRetrievalEvent
|
+--**jp.or.arib.tv.si.SITableUpdatedEvent**

Mounted interface for all:

java.io.Serializable

public class **SITableUpdatedEvent**
extends SIRetrievalEvent

This event is sent as a response when the table that transmits information about the target object for the SI acquisition request is updated and the descriptor information that conforms to the old object is not available. In this case, the application should update the SIInformation object at first. Then the information about the descriptor should be requested again.

Reference:

SIRetrievalListener, java.io.Serializable

General information on the constructor
SITableUpdatedEvent(java.lang.Object appData, SIRequest request) Constructor

Method inherited from Class jp.or.arib.tv.si.SIRetrievalEvent
getAppData, getSource

Method inherited from Class java.util.EventObject
toString

Method inherited from Class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Details of the constructor

SITableUpdatedEvent
public **SITableUpdatedEvent**(java.lang.Object appData,
SIRequest request)

Constructor

Parameter:

appData - Application data given when the request method is called.
request – SIRequest instance that is the origin of the event

M.2.35 SIUtil
jp.or.arib.tv.si
Class SIUtil

java.lang.Object
  |
  +--**jp.or.arib.tv.si.SIUtil**

public class **SIUtil**
extends java.lang.Object

Class that includes utility function relevant to SI

General information on the method	
static java.lang.String	**convertSIStringToJavaString**(byte[] aribSIText, int offset, int length) This method converts coded text string to string object of Java based on ARIB STD-B10 Part 2, Appendix A.

Method inherited from Class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Details of the method

convertSIStringToJavaString
public static java.lang.String **convertSIStringToJavaString**(byte[] aribSIText,
  int offset,
  int length)
  throws SIIllegalArgumentException

This method converts coded text string to string object of Java based on ARIB STD-B10 Part 2 Appendix A.

The text to be converted is included between index value ‘offset’ of 'aribSIText' array and 'offset+length-1'.

In case the target section is not codec appropriately based on ARIB STD-B10 Part 2 Appendix A, the result is undefined.

Parameter:
aribSIText – Byte array that includes the string of conversion target.
offset – Offset that indicates the start location of ARIB-SI text included in aribSIText.
length – ARIB-SI text length (the number of bytes).
Return value:
Converted text
Exception:
SIIllegalArgumentException – In case the value of offset or offset+length-1 is false as

index value of aribSIText.

M.2.36 SIException
jp.or.arib.tv.si
Class SIException

java.lang.Object
  |
  +--java.lang.Throwable
  |
  +--java.lang.Exception
  |
  +--**jp.or.arib.tv.si.SIException**

Mounted interface for all:
java.io.Serializable

Known sub class of direct line :
SIIllegalArgumentException, SIInvalidPeriodException

public abstract class **SIException**
extends java.lang.Exception

This class is a root of the SI exception hierarchy structure.

Reference:
java.io.Serializable

General information on the constructor
SIException() Default constructor of this exception
SIException(java.lang.String reason) Constructor of the SI exception that has a reason to be designated.

Method inherited from Class java.lang.Throwable
fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Method inherited from Class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Details of the constructor

SIException
public **SIException**()
Default constructor of the exception

SIException
public **SIException**(java.lang.String reason)
Constructor of the SI exception that has a reason to be designated.

M.2.37 SIIllegalArgumentException
jp.or.arib.tv.si
Class SIIllegalArgumentException

java.lang.Object
|
+--java.lang.Throwable
|
+--java.lang.Exception
|
+--jp.or.arib.tv.si.SIException
|
+--**jp.or.arib.tv.si.SIIllegalArgumentException**

Mounted interface for all:
java.io.Serializable

public class **SIIllegalArgumentException**
extends SIException

This exception arises when more than one inappropriate argument is given (e.g. numeric values outside the scope).

Reference:
java.io.Serializable

General information on the constructor
SIIllegalArgumentException()

Default constructor of the exception
SIIllegalArgumentException(java.lang.String reason) Constructor of the exception that has reason for being specified.

Method inherited from Class java.lang.Throwable
fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Method inherited from Class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Details of the constructor

SIIllegalArgumentException

public **SIIllegalArgumentException**()

Default constructor of this exception

M.2.38 SIInvalidPeriodException

jp.or.arib.tv.si

Class SIInvalidPeriodException

java.lang.Object
 |
 +--java.lang.Throwable
 |
 +--java.lang.Exception
 |
 +--jp.or.arib.tv.si.SIException
 |
 +--**jp.or.arib.tv.si.SIInvalidPeriodException**

Mounted interface for all:

java.io.Serializable

public class **SIInvalidPeriodException**
extends SIException

This exception arises when the specified time span is inappropriate (e.g. the start time is later than the end time).

Reference:

java.io.Serializable

General information on the constructor
SIInvalidPeriodException() Exceptional default constructor
SIInvalidPeriodException(java.lang.String reason) Exceptional constructor that has a reason for being specified.

Method inherited from Class java.lang.Throwable
fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Method inherited from Class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Details of the constructor

SIInvalidPeriodException

public **SIInvalidPeriodException**()

Default constructor of exception

SIInvalidPeriodException

public **SIInvalidPeriodException**(java.lang.String reason)

Constructor of exception that has a reason for being specified.

Annex N Streamed media API extensions

This Annex conforms to GEM1.0”Annex N: Streamed media API extensions”. In addition, there are amendments and/or additions as follows.

The following class and interfaces are added as API to detect changes of audio language.

jp.or.arib.tv.AudioLanguageEventControl
jp.or.arib.tv.AudioLanguageEventListener
jp.or.arib.tv.AudioLanguageChangedEvent

This standard does not apply the resolution 720 × 576 described in MHP1.0 “Annex N: Streamed media API extensions” but conforms to the resolution described in ARIB STD-B21 for video display resolution. In addition, jp.or.arib.tv.media.ARIBVideoFormatControl interface is additionally defined.

N.1 Package jp.or.arib.tv.media

ARIB-specific extension to the Java Media Framework

General information on the interface	
ARIBVideoFormatControl	This interface provides a method to acquire ARIB-specific information relevant to the currently displayed video.
AudioLanguageEventControl	This interface provides a function to validate and/or invalidate the monitoring of audio language change events.
AudioLanguageEventListener	This is the listener that is notified of audio language changes.

General information on the class	
AudioLanguageChangedEvent	This in an event to notify of main audio language changes.

N.1.1 Explanation of package jp.or.arib.tv.media

ARIB-specific extensions to the Java Media Framework

N.1.2 ARIBVideoFormatControl

jp.or.arib.tv.media

Interface ARIBVideoFormatControl

Super interface for all:

org.dvb.media.VideoFormatControl

public interface **ARIBVideoFormatControl**
extends org.dvb.media.VideoFormatControl

The interface provides a method to acquire ARIB-specific information relevant to the currently displayed video.

Fields inherited from Interface org.dvb.media.VideoFormatControl
AFD_14_9, AFD_14_9_TOP, AFD_16_9, AFD_16_9_SP_14_9, AFD_16_9_SP_4_3, AFD_16_9_TOP, AFD_4_3, AFD_4_3_SP_14_9, AFD_GT_16_9, AFD_NOT_PRESENT, AFD_SAME, ASPECT_RATIO_16_9, ASPECT_RATIO_2_21_1, ASPECT_RATIO_4_3, ASPECT_RATIO_UNKNOWN, DAR_16_9, DAR_4_3, DFC_PLATFORM, DFC_PROCESSING_CCO, DFC_PROCESSING_FULL, DFC_PROCESSING_LB_14_9, DFC_PROCESSING_LB_16_9, DFC_PROCESSING_LB_2_21_1_ON_16_9, DFC_PROCESSING_LB_2_21_1_ON_4_3, DFC_PROCESSING_NONE, DFC_PROCESSING_PAN_SCAN, DFC_PROCESSING_UNKNOWN

General information on the method	
java.awt.Dimension	**getDisplayVideoSize**() This method acquires the size of the display area specified by sequence_display_extension.
int	**getProgressiveSequence**() This method acquires the progressive_sequence specified by sequence_extension.
java.awt.Dimension	**getSourceVideoSize**() This method acquires the size of the video source specified by sequence_header.

Methods inherited from Interface org.dvb.media.VideoFormatControl
addVideoFormatListener, getActiveFormatDefinition, getAspectRatio, getDecoderFormatConversion, getDisplayAspectRatio, getVideoTransformation, isPlatform, removeVideoFormatListener

Details of the method

getDisplayVideoSize

public java.awt.Dimension **getDisplayVideoSize**()

This method acquires the size of the display area specified by sequence_display_extension.

Return value:

Dimension object that stores the values of display_vertical_size and display_horizontal_size. If the display area is not specified by sequence_display_extension, the value is “null”.

getProgressiveSequence

public int **getProgressiveSequence**()

This method acquires the progressive_sequence specified by sequence_extension.

Return value:

progressive_sequence value

getSourceVideoSize

public java.awt.Dimension **getSourceVideoSize**()

This method acquires the size of video source specified by sequence_header.

Return value:

Dimension object that stores the values of vertical_size and horizontal_size.

N.1.3 AudioLanguageEventControl

jp.or.arib.tv.media

Interface AudioLanguageEventControl

Super interface for all:

org.davic.media.AudioLanguageControl, org.davic.media.LanguageControl

public interface **AudioLanguageEventControl**
extends org.davic.media.AudioLanguageControl
The interface provides a function to validate and/or invalidate the monitoring of audio language change events.

General information on the method	
void	**addAudioLanguageEventListener**(AudioLanguageEventListener l) This method adds the listener to monitor audio language changes.
void	**removeAudioLanguageEventListener**(AudioLanguageEventListener l) This method removes the listener that monitors audio language changes.

Method inherited from Interface org.davic.media.LanguageControl
getCurrentLanguage, listAvailableLanguages, selectDefaultLanguage, selectLanguage

Details of the method

addAudioLanguageEventListener
public void **addAudioLanguageEventListener**(AudioLanguageEventListener l)

This method adds the listener to monitor audio language changes.

Parameter:

l – listener to monitor the event

removeAudioLanguageEventListener
public void **removeAudioLanguageEventListener**(AudioLanguageEventListener l)

This method removes the listener that monitors audio language changes.

Parameter:

l – listener to be removed

N.1.4 AudioLanguageEventListener
jp.or.arib.tv.media
Interface AudioLanguageEventListener

Super interface for all:

java.util.EventListener

public interface **AudioLanguageEventListener**

extends java.util.EventListener

Listener that is notified of audio language changes.

General information on the method	
void	**audioLanguageChanged**(AudioLanguageChangedEvent event) This method is called when the main audio language is changed.

Details of the method

audioLanguageChanged

public void **audioLanguageChanged**(AudioLanguageChangedEvent event)

This method is called when the main audio language is changed.

Parameter:

event – Generated event

N.1.5 AudioLanguageChangedEvent

jp.or.arib.tv.media

Class AudioLanguageChangedEvent

java.lang.Object
 |
 +--java.util.EventObject
 |
 +--**jp.or.arib.tv.media.AudioLanguageChangedEvent**

Mounted interface for all:

java.io.Serializable

public class **AudioLanguageChangedEvent**
extends java.util.EventObject
Event to notify of the main audio language is changed.

Reference:

java.io.Serializable

General information on the method	
java.lang.String	**getSelectedLanguage**() This method acquires the language that is newly set for the main audio.

Method inherited from Class java.util.EventObject
getSource, toString

Method inherited from Class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Details of the method

getSelectedLanguage

public java.lang.String **getSelectedLanguage**()

This method acquires the language that is newly set for the main audio.

Return value:

Language code specified by ISO 639 standard.

Annex O Integration of the Java TV SI API

O.1 Outline

While GEM1.0 adopts Java TV SI API as abstract SI access API that is not present in the network protocol, each SI system and the integration is to be specified by each standardization organization. Requirements shall conform to GEM1.0 “Annex O: Integration of the JavaTV SI API”. This chapter describes the data structure of service information specified in ARIB STD-B10, mapping with JavaTV SI API and the integration of ARIB-SI API specified in Annex M “SI Access API” with JavaTV SI API.

O.2 Mapping of Java TV SI to ARIB-SI

This section describes how all relevant interfaces and methods on JavaTV SI API are mapped to ARIB service information.

Even if a listener is added to monitor the changes on the SI table or of data to be transmitted by the SI table, the event is not generated for the current version of the table (or data) detected on the network at the time of the listener addition. In other words, events are generated only for the changes after the detection of the current version.

O.2.1 javax.tv.service.Service

This Service interface indicates not only the entries on the service list accumulated in the receiver but also the service entries on the network and has not been accumulated in the receiver.

As for the mounting status of receivers, some objects that mount this interface may and may not mount the ServiceNumber interface. In addition, some objects mount the Service interface only and others mount the ServiceNumber interface as well despite having the same receiver mounting status.

Note)
As described in GEM1.0 “14.9 Service identification,” Java TV can apply two types of locator for service identification. One is dependent on the transmission protocol and the other is not dependent on the transmission protocol. This standard applies only the latter one that is not dependent on the transmission protocol.

Mapping of each method is shown below.

O.2.1.1 getName

This returns service names. Some mounting status allows end users to compile the names

according to the preference of users. If the contents of the field are acquired from the service information in default via the terminal, the ARIB-AE terminal uses the character information field of the service name of the service descriptor.

The character string information in SI is coded according to the 8bit-code specified in ARIB STD-B10 “Annex A Character Code”. When it is called according to the Java method, it shall be acquired as appropriate character string information of Unicode.

O.2.1.2 getServiceType
This returns the JavaTV service type according to the mapping defined in “O.2.3 javax.tv.service.ServiceType”.

O.2.2 javax.tv.service.navigation.ServiceComponent
ServiceComponent interface provides information included in the component descriptor, audio component descriptor and data contents descriptor of EIT.

O.2.2.1 getComponentName
If the language code returned by javax.tv.service.SIManager.getPreferredLanguage corresponds to the audio component descriptor or the data contents descriptor, a description based on the language code is returned. If not, any description available in EIT is returned and the selection of description depends on the mounting status.

O.2.2.2 getAssociatedLanguage
This returns the language code to indicate the language of the component as specified in ISO 639-2 based on the component descriptor, audio component descriptor and data contents descriptor. The language is not necessarily the one selected for the name acquired by getName().

O.2.2.3 getStreamType
Based on “O.2.4 javax.tv.service.navigation.StreamType,” the stream type is returned according to the mapping of JavaTV stream type.

O.2.3 javax.tv.service.ServiceType
ARIB-SI service type is specified by the service type listed in Table 6-25 of ARIB STD-B10. Corresponding status to the Java TV service type is shown below.

Table O-1 Correspondence of the JavaTV Service Type to ARIB

ARIB Service type ARIB	Description of ARIB service type ARIB	Java TV service type
0x00	Undefined	UNKNOWN
0x01	Digital TV service	DIGITAL_TV
0x02	Digital audio service	DIGITAL_RADIO
0x03-0x7F	Undefined	UNKNOWN
0x80-0xA0	Broadcasters definition	UNKNOWN
0xA1	Temporary video service	DIGITAL_TV
0xA2	Temporary audio service	DIGITAL_RADIO
0xA3	Temporary data service	DATA_BROADCAST
0xA4	Engineering download service	UNKNOWN
0xA5	Promotion video service	DIGITAL_TV
0xA6	Promotion audio service	DIGITAL_RADIO
0xA7	Promotion data service	DATA_BROADCAST
0xA8	Data service of advance accumulation	DATA_BROADCAST
0xA9	Data service for accumulation only	DATA_BROADCAST
0xAA-0xBF	Undefined (in the scope of definition by standard organization)	UNKNOWN
0xC0	Data service	DATA_BROADCAST
0xC1-0xFF	Undefined	UNKNOWN

O.2.4 javax.tv.service.StreamType

The streamARIB-SI component contents and component types are specified as in Table 6-5 of ARIB STD-B10. The correspondence to Java TV is shown below.

Table O-2 Correspondence of the Java TV stream types to ARIB component contents and component types

Component contents	Component type	JavaTV stream type Java
0x00	0x00-0xFF	UNKNOWN
0x01	0x00-0xFF	VIDEO
0x02	0x00-0xFF	AUDIO
0x03-0x0B		UNKNOWN (Specified in operation guideline separately)
0x0C-0x0F	0x00-0xFF	UNKNOWN

In case the data component descriptor, instead of the component descriptor or the audio component descriptor, is associated with the specified component, the following policies shall be followed.

- If access is available to make transmitting contents a file system by a multi media coding system using a data carousel/object carousel, the type should be DATA.
- If access is not available to make transmitting contents a file system by a multi media coding system using a data carousel/object carousel, the type should be

SECTION.

- If a private section transmission protocol is used, the type should be SECTION.
- For other cases that the transmission protocol cannot be identified on JavaTV, the type is UNKNOWN.

According to the policies above, the corresponding status to the data coding scheme ID that is currently specified is shown for reference in Table O-3 below.

Table O3 Correspondence of the data coding scheme ID and Java TV stream type (informative)

Data coding scheme ID (data_component_id)	Data coding scheme	Java TV stream type
0x0000…0x0006		UNKNOWN
0x0007	ARIB-XML base multi media coding scheme	DATA
0x0008	ARIB-closed caption / teletext coding scheme	SUBTITLES
0x0009	ARIB- data download system	SECTIONS
0x000A	G-Guide Gold	SECTIONS
0x000B	BML for east longitude 110 degrees CS	DATA
0x000C	Multimedia coding scheme for digital terrestrial broadcasting (A profile)	DATA
0x000D	Multimedia coding scheme for digital terrestrial broadcasting (C profile)	DATA
0x000E	Multimedia coding scheme for digital terrestrial broadcasting (P profile)	DATA
0x000F	Multimedia coding scheme for digital terrestrial broadcasting (E profile)	DATA
[T.B.D]	ARIB-program index coding scheme	SECTIONS
[T.B.D]	ARIB-J application coding scheme	DATA
[T.B.D]	ARIB-AIT coding scheme	SECTIONS
Other than those above (- 0xFFFF)		UNKNOWN

For the ID values that are not specified in ARIB STD-B10, the correspondence of the Java TV stream to the data coding scheme ID shall be UNKNOWN.

O.2.5 javax.tv.service.SIElement

This interface is to be mounted on the object that mounts the interface such as Network, Bouquet, TransportStream, ServiceDetails, ServiceComponent and ProgramEvent.

O.2.5.1 getServiceInformationType

This method returns UNKNOWN as specified by javax.tv.service.ServiceInformationType. [1]

[1] At the time of this standard establishment, the constant corresponding Java TV API 1.0 to express SI in ARIB has not been specified. Accordingly the value UNKNOWN is returned. However, it is expected to introduce a constant that express SI specified in ARIB. The reference standard of Java TV for this standard is publicly known as 1.0 currently. It is preferable for the definition to be revised in accordance with the Java TV revision in the future.

O.2.6 javax.tv.service.SIManager

O.2.6.1 getSupportedDimensions

The Parental rating descriptor defined in ARIB STD-B10 specifies the rating based on age. In order to indicate the rating defined by ARIB, getSupportedDimensions returns the array that includes the string “ARIB Age based rating”.

O.2.6.2 getRatingDimension

If the character string “ARIB Age based rating” is assigned as an argument this method returns the object mounting RatingDimension interface referred to in “O.2.10 javax.tv.service.RatingDimension”.

O.2.6.3 retrieveSIElement

If a locator that indicates the service is given, the object mounting ServiceDetails interface is returned. A locator that indicates the program event is not supported and this method fails with SIRequestFailureType(INSUFFICIENT_RESOURCES) accordingly.
Other types of locators are supported as defined.

Note)
Nevertheless, a program event can be acquired at the specified time using javax.tv.service.guide.ProgramSchedule.

O.2.6.4 getTransports

The object to be returned by this method mounts the Transport interface referred in “O.2.12 javax.tv.service.transport.Transport”.

O.2.6.5 filterServices

Service filtering is supported with ServiceFilters. To support this, SIElementFilter defined in “O.2.7 javax.tv.service.nabigation.SIElementFilter” is required.

O.2.6.6 retrieveProgramEvent

This method fails with SIRequestFailureType (INSUFFICIENT_RESOURCES).

Note)
Nevertheless, a program event can be acquired at the specified time using javax.tv.service.guide.ProgramSchedule.

O.2.7 javax.tv.service.navigation.SIElementFilter

The SIElementFilter recognizes the filtering of SIElement-based services. This filter is supported for the objects of Network and TransportStream. The constructor goes through

FilterNotSupportedException for other SIElement objects.

O.2.8 javax.tv.service.navigation.ServiceDetails
The ServiceDetails interface indicates the information relevant to the service acquired from Service Information. The objects mounting the interface mount CAIdentification interface according to the mapping defined in “O.2.9 javax.tv.service.navigation.CAIdentification”. These objects do not mount the ServiceNumber interface.

O.2.8.1 getLongName
The service name stored in the character information field of the service name of the service descriptor is returned. The character string information in SI is coded according to the 8bit-code specified in ARIB STD-B10 “Annex A Character Code”. When it is called according to the Java method, it shall be acquired as appropriate character string information of Unicode.

O.2.8.2 getServiceType
JavaTV service type is returned according to the mapping defined in “O.2.3 javax.tv.service.ServiceType”.

O.2.8.3 retrieveServiceDescription
In ARIB, service information does not include this kind of service descriptor. This method always calls notifyFailure of the SIRequest object called by the SIReqestFailureType of DATA_UNAVAILABLE .

O.2.8.4 retrieveComponents
Information relevant to ServiceComponents is acquired by the component descriptor, audio component descriptor and data contents descriptor in EIT[Present/Following] that has corresponding language code to the language code to be returned by javax.tv.service.SIManager.getPreferredLanguage. If the corresponding language code is not present, usable information is acquired from any of the above-mentioned descriptors but the selection criteria depends on the mounting status.

O.2.9 javax.tv.service.navigation.CAIdentification
This interface is to be mounted on the object that mounts the interfaces of ServiceDetails, ProgramEvent or Bouquet.

O.2.9.1 getCASystemIds
The array of the integer that indicates the conditional access system (CA_system_id) is returned from the CA ID descriptor. If the CA ID descriptor is not present, an empty array is returned.

O.2.9.2 isFree
If the interface is mounted on the object that mounts ServiceDetails or ProgramEvent, this method returns “true” only when “0” is set for the free_CA_mode field of the entry of SDT or EIT.
If the interface is mounted on the object that mounts the Bouquet interface, this method returns “true” only when the CA ID descriptor is present on BAT.

O.2.10 javax.tv.service.RatingDimension
The parental rating descriptor indicates the rating based on age, which can be extended in accordance with other restrictions. In ARIB STD-B10, rating of 15 levels from 4 to 18 years old is defined specifying the recommended lowest age.
The object that indicates the rating is required to mount the following method.

O.2.10.1 getDimensionName
Character string “ARIB Age based rating” is returned.

O.2.10.2 getNumberOfLevels
The integer “15” is returned.

O.2.10.3 getRatingLevelDescription
The two character strings shown below are returned,
{“Over n,” “Recommended minimum age:n years”}
where “n” means entry parameter + 4(years old).

O.2.11 javax.tv.service.navigation.ServiceProviderInformation
This interface is to be mounted on the object that mounts the ServiceDetails interface.

O.2.11.1 getProviderName
The broadcaster name stored in the service descriptor in SD is returned.

O.2.12 javax.tv.service.transport.Transport
The object mounting Transport interface needs to mount the NetworkCollection and the BouquetCollection as well.

O.2.13 javax.tv.service.transport.Bouquet
The Bouquet interface is mounted by the object that indicates the bouquet in ARIB Service Information. The object mounting this interface needs to mount the CAIdetification interface specified in “O.2.9 javax.tv.service.navigation.CAIdentification” as well.

O.2.13.1 getBouquetID
The integer is returned for the bouquet ID value.

O.2.13.2 getName
The bouquet name stored in the bouquet name descriptor of BAT is returned.

O.2.13.3 getLocator
The mounting status dependent javax.tv.locator.Locator object is returned. This does not have standardized external expression. Therefore, it is not necessarily jp.or.arib.tv.net.ARIBLocator.

O.2.14 javax.tv.service.transport.Network
The network interface is mounted on the object that indicates the network of ARIB Service Information.

O.2.14.1 getNetworkID
An integer value is returned for the network ID value.

O.2.14.2 getName
The network name stored in network name descriptor of NIT is returned.

O.2.14.3 getLocator
The mounting status dependent javax.tv.locator.Locator is returned. This does not have standardized external expression. Therefore, it is not necessarily jp.or.arib.tv.net.ARIBLocator.

O.2.15 javax.tv.service.transport.TransportStream
The TransportStream interface is mounted on the object that indicates the transport stream.

O.2.15.1 getTransportStreamID
The transport stream ID value is returned by the integer value.

O.2.15.2 getDescription
ARIB Service Information does not have a transport stream description. Therefore, this method returns an empty character string.

O.2.16 javax.tv.service.guide.ProgramEvent
This interface is mounted on the object that indicates the program event. The object mounting this interface needs to mount the CAIdetification interface specified in “O.2.9 javax.tv.service.navigation.CAIdentification” as well.

O.2.16.1 getDuration
The value of the duration (duration) is returned from the entry of the program event in EIT.

O.2.16.2 getStartTime
The value of the start time (start_time) is returned from the entry of the program event in EIT.

O.2.16.3 getEndTime
The end time is calculated and returned according to the duration and the start time of the program event in EIT.

O.2.16.4 getName
The program name stored in the short form event descriptor in EIT that corresponds to the language code to be acquired by javax.tv.service.SIManager.getPreferredLanguage is returned. If the corresponding language is not present, any usable information among the above-mentioned descriptors is acquired and the selection of criteria depends on the mounting status.

O.2.16.5 retrieveDescription
The program name stored in the short form event descriptor in EIT that corresponds to the language code to be acquired by javax.tv.service.SIManager.getPreferredLanguage is returned. If the corresponding language is not present, any usable information among the above-mentioned descriptors is acquired and the selection of criteria depends on the mounting status.

The character string to be acquired by the retrieveDescription method shall be converted into the corresponsive Unicode character string. APD (active position down) code 0x0a is mapped to the Java line feed code ’¥n’ and APR (active position line forward) code is mapped to ’¥r’. Both are treated as a line end symbol.

The character string acquired by getProgramEventDescrption() of ProgramEventDescription object is given as a character string that includes the program description.

O.2.16.6 getRating
The rating from the parental rating descriptor is returned according to the mapping defined in “O.2.17 javax.tv.service.guide.ContentRatingAdvisory”.

O.2.17 javax.tv.service.guide.ContentRatingAdvisory

O.2.17.1 getDimensionNames
The array that includes the character string “ARIB Age based rating” is returned.

O.2.17.2 getRatingLevel
When the “ARIB Age based rating” is assigned as the parameter and the rating value of the parental rating descriptor is between 0x01 and 0x0F, the value to be returned is “1” subtracted from the value in the descriptor. In other cases, “-1” is returned.

O.2.17.3 getRatingText
When the “ARIB Age based rating” is assigned as the parameter and the rating value of the parental rating descriptor is between 0x01 and 0x0F, the character string “Recommended minimum age: n years” is returned. The value added “3” to the rating value in the descriptor makes ‘n’ to be stored. In other cases, this method returns an empty character string.

Note)
If the information needs to be indicated in a language other than English in this application, the rating value should be acquired by the getRatingLevel() method and presented to the audience by its own character coding system.

O.2.17.4 getDisplayText
If the rating value of the parental rating descriptor is between 0x01 and 0x0F, the character string that includes “Over n” is returned. The value adding “3” to the rating value in the descriptor makes ‘n’ to be stored.
Note)
If the information needs to be indicated in a language other than English in this application, the rating value should be acquired by the getRatingLevel() method and presented to the audience by its own character coding system.

O.3 Integration of Java TV SI API and ARIB-SI API
Java TV SI interfaces corresponding to the objects mounting ARIB-SI API interfaces shall be also provided. At the same time, corresponding ARIB-SI API interfaces shall be mounted so that SI objects can be acquired by Java TV API.

The corresponding status for each object of both AIP interfaces is shown below.

Table O-4 Corresponding status between Java TV SI API and ARIB-SI API

jp.or.arib.tv.si.SINetwork	javax.tv.service.transport.Network
jp.or.arib.tv.si.SIBouquet	javax.tv.service.transport.Bouquet
jp.or.arib.tv.si.SITransportStreamNIT	javax.tv.service.transport.TransportStream

jp.or.arib.tv.si.SIService	javax.tv.service.navigation.ServiceDetails
jp.or.arib.tv.si.SIEvent	javax.tv.service.guide.ProgramEvent
jp.or.arib.tv.si.SIBroadcaster	No corresponding interface

Annex P Broadcast transport protocol access

P.1 Outline

This Annex specifies the API of the org.dvb.dsmcc package.

The mapping of the org.dvb.dsmcc package to the API defined in GEM1.0 “Annex P: Broadcast transport protocol access” is specified in the case of using the DSM-CC data carousel and the event message protocols defined in ARIB STD-B 24 Part 3.

In the case of using the object carousel as transport protocol, this Annex conforms to GEM1.0 “Annex P: Broadcast transport protocol access,” except for the special notes in this Annex.

P.2 org.dvb.dsmcc package

The additional definitions described below to the GEM 1.0 Annex P P.2 org.dvb.dsmcc section realizes the mapping to the DSM-CC data carousel as well as the event message defined in ARIB-STD B24 Part 3.

P.2.1 DSMCCObject

Each object of file, stream and stream event as resources in the module is expressed in the DSM-CC data carousel to be used in this standard. As for directories, the entity of an object is not present like the object carousel. Therefore, this standard defines the directory objects as follows.

(1) Root directory of service domain
This is identified by the following:
arib-dc://<original_network_id>.<transport_stream_id>.<service_id>[;<content_id>][.<event_id>]/<component_tag>

The directory information is associated with the module information of the associated carousel. When this directory object is loaded, the DII module information is analyzed to acquire the object list right under the said directory.

(2) Module hierarchy directory
This is identified by the following:
arib-dc://<original_network_id>.<transport_stream_id>.<service_id>[;<content_id>][.<event_id>]/<component_tag>/<moduleName>

The directory information is associated with the source list for ARIB-J included in the said module specified in appendix B 2.1.1.

When this directory object is loaded, all resource lists shall be analyzed to acquire the object list under the said directory. At this time, the whole hierarchical structure under the said directory is clarified but it depends on the mounting status whether the directory object status under the directory is loaded (Loaded) or not.

Note that file the file object is identified by;
arib-dc://<original_network_id>.<transport_stream_id>.<service_id>[;<content_id>][.<event_id>]/<component_tag>/<moduleName>
where resources are mapped to the module directly.

(3) Directory of hierarchy corresponding to "/" in the resource name
This is identified by the following:
arib-dc://<original_network_id>.<transport_stream_id>.<service_id>[;<content_id>][.<event_id>]/<component_tag>/<moduleName>*(/<directoryName>)

The directory information is associated with the resource list for ARIB-J that is included in the module to which the associated path belongs.

When this directory object is loaded, the whole resource lists shall be analyzed to acquire the object list right under the said directory. At this time, the whole hierarchical structure under the said directory is clarified but it depends on the mounting status whether the directory object status under the directory becomes loaded (Loaded) or not. In addition, the upper directory of the said directory is automatically loaded but it depends on the mounting status whether the status of the directory object under the upper directory becomes loaded (Loaded) or not.

When each object of file, stream and stream event is loaded, the upper directory of the associated object is also automatically loaded and it depends on the mounting status whether the status of the directory object under the upper directory becomes loaded (Loaded) or not.

P.2.1.1 addObjectChangeEventListener()
When ES is eliminated by the PMT update, ObjectChangeEvent is not generated.

When a data event is updated, ObjectChangeEvent relevant to the root directory object of the corresponding service domain is generated because transaction_id is updated at the same time. At this time, the service domain is detached. (This can be confirmed by isAttached().) In addition, ObjectChangeEvent relevant to other DSM-CC objects than the root directory object is not generated.

If a module is excluded from the DII module due to the DII update, ObjectChangeEvent

relevant to the DSM-CC object that is associated with the said module is not generated.

P.2.1.2 asynchronousLoad()
When this function is called for the directory object that corresponds to "/" in the resource name, SuccessEvent is generated if one or more resources include the associated path in the resource information of the module. On the other hand, InvalidPathnameException or InvalidPathnameEvent is generated if there are no resources that include the associated path.

P.2.1.3 getSigners()
The method of signing for the object shall conform to the policy of "Chapter 12 Security" of this standard.

P.2.1.4 isLoaded()
If the directory object corresponding to the parent directory of the said object has been loaded and the said object is included in the said directory, "true" is returned.

When this method is called for the root directory object of the service domain, "true" is returned if the said directory object has been loaded.

P.2.1.5 isStream()
When isObjectKindKnown() for the said object is "true" and stream_flag in the resource information is "1," "true" is returned.

P.2.1.6 isStreamEvent()
When isObjectKindKnown() for the said object is "true," and stream_flag and stream_event_flag in the resource is "1," "true" is returned.

P.2.1.7 loadDirectoryEntry(AsynchronousLoadingEventListener)
When the parent directory of the said object corresponds to "/" in resourceName, SuccessEvent is generated if at least one or more resources include the path of the said directory. On the other hand, InvalidPathnameException or InvalidPathnameEvent is generated if there are no resources that include the said path.

If this method is called for the root directory object of the service domain, InvalidPathNameException is generated.

P.2.1.8 synchronousLoad()
When this function is called for the directory object that corresponds to "/" in resourceName, InvalidPathnameException is generated if there are no resources that include the associated path in the resource information of the module.

P.2.2 DSMCCStream

P.2.2.1 addNPTListener(NPTListener)

The listener for the event relevant to the NPT reference descriptor to be identified by npt_component_tag of dsm_contentId in the resource information corresponding to the said stream is added.

P.2.2.2 getDuration()

The duration value in the resource information specified in Annex B is returned.

P.2.2.3 getStreamLocator()

The interpretation of the acquired locator is specified in Operation separately.

P.2.2.4 isAudio()

If audio_flag is “1” in the resource information specified in Annex B, “true” is returned.

P.2.2.5 isData()

If data_flag is “1” in the resource information specified in Annex B, “true” is returned.

P.2.2.6 isMPEGProgram()

If data carousel is used, “false” is always returned.

P.2.2.7 isVideo()

If video_flag is “1” in the resource information specified in Annex B, “true” is returned.

P.2.3 NPTDiscontinuityEvent

The definition of NPT discontinuity is specified in Operation separately.

P.2.4 NPTRemoveEvent

It is generated when the transmission of the NPT reference descriptor fails. The judgment criteria to determine that the NPT reference descriptor is not transmitted are specified in Operation separately.

P.2.5 ObjectChangedEvent

For the resource version, the version of the module that includes the said resource is applied. In addition, the version of the module is also applied for the directory object corresponding to “/” in resourceName. Therefore, this event is generated for all objects relevant to the said module when even one resource in the module is updated.

Service Gateway is not transmitted as a module in the data carousel. Accordingly, transaction_id in the DDI of the said carousel is used for the version number of the root directory of the service domain. Therefore, ObjectChangeEvent is generated to the root directory object whenever the DII is updated.

P.2.6 ServerDeliveryErrorEvent
Because DSM-CC U-U protocol is not used via the return channel, this event is not generated under this standard.

P.2.7 ServerDeliveryException
Because DSM-CC U-U protocol is not used via the return channel, this exception is not generated under this standard.

P.2.8 ServiceDomain
P.2.8.1 attach(byte[])
NSAP address specified in Annex B.111 is used.

P.2.8.2 attach(Locator)
The Locator of Service Domain specified in 14.8 is used.

P.2.8.3 attach(Locator, int)
The Locator for the Service specified in 14.8. is used for aDVBService. In addition, downloadId of the said carousel is specified for aCarouselId.

P.2.8.4 getLocator()
The Locator of Service Domain specified in 14.8 is returned.

P.2.9 ServiceXFRErrorEvent
In data carousel, the function corresponding to LiteOptionProfileBody in the object carousel is not used. Therefore this event is not generated.

P.2.10 ServiceXFRErrorException
In the data carousel, the function corresponding to LiteOptionProfileBody in the object carousel is not used. Therefore this exception is not generated.

P.2.11 ServiceXFRReference
In the data carousel, the function corresponding to LiteOptionProfileBody in the object carousel is not used. Therefore this service is not used.

P.2.12 StreamEvent
P.2.12.1 StreamEvent(DSMCCStreamEvent, long, String, int, byte[])
The values shown below are assigned for each argument.

npt: In the general-purpose event message descriptor, if time_mode is 0x02, the value is of event_msg_NPT(33bit). If time_mode is 0x00, the value is “-1”.
name: eventName stored in the resource information specified in the Annex
eventId: event_msg_id (16bit) of the general-purpose event message.
eventData: private_data of the general event message descriptor.

Annex Q Datagram socket buffer control

This Annex defines the extended API of the relevant UDP network function and conforms to GEM1.0 “Annex Q: Datagram Socket Buffer Control”.

Annex R ARIB-J return channel connection management API

This Annex defines API relevant to Return Channel Connection Management and conforms to “Annex R: DVB-J Return Channel Connection Management API”.

In addition, jp.or.arib.tv.net.rc.ARIBRCInterface is additionally specified in order to add the definition and acquisition method of the ARIB-specific Return Channel Type.

R.1 Package jp.or.arib.tv.net.rc
Definition of ARIB-specific return channel type

General information on the interface	
ARIBRCInterface	An interface for definition and acquisition of ARIB-specific return channel type.

R.1.1 Explanation of package jp.or.arib.tv.net.rc
Definition of ARIB-specific return channel type.

R.1.2 ARIBRCInterface
jp.or.arib.tv.net.rc
Interface ARIBRCInterface

public interface **ARIBRCInterface**

This is an interface to define and acquire the ARIB-specific return channel type. In the ARIB-AE environment, the objects to be acquired by the methods getInterface and getInterfaces of org.dvb.net.rc.RCInterfaceManager definitely mount this interface.

Reference:

org.dvb.net.rc.RCInterface, org.dvb.net.rc.RCInterfaceManager

General information on the field	
static int	**ARIB_TYPE_ETHERNET_DHCP** ARIB-specific return channel type: Ethernet (DHCP)
static int	**ARIB_TYPE_ETHERNET_FIXED_IP** ARIB-specific return channel type: Ethernet (Fixed IP)
static int	**ARIB_TYPE_ETHERNET_PPPOE** ARIB-specific return channel type: Ethernet (PPPoE)
static int	**ARIB_TYPE_MOBILE_PHONE** ARIB-specific return channel type: mobile phone (In case type information is not available.)
static int	**ARIB_TYPE_MOBILE_PHONE_CDMA_CELLUARSYSTEM** ARIB-specific return channel type: mobile phone (CDMA CelluarSystem)
static int	**ARIB_TYPE_MOBILE_PHONE_DS_CDMA** ARIB-specific return channel type: mobile phone (DS-CDMA)
static int	**ARIB_TYPE_MOBILE_PHONE_MC_CDMA** A ARIB-specific return channel type: mobile phone (MC-CDMA)
static int	**ARIB_TYPE_MOBILE_PHONE_PDC** ARIB-specific return channel type: mobile phone (PDC)
static int	**ARIB_TYPE_MOBILE_PHONE_PDCP** ARIB-specific return channel type: mobile phone (PDC-P)
static int	**ARIB_TYPE_PHS**

	ARIB-specific return channel type: PHS (In case type information is not available.)
static int	**ARIB_TYPE_PHS_PIAFS20** ARIB-specific return channel type: PHS (PIAFS2.0)
static int	**ARIB_TYPE_PHS_PIAFS21** ARIB-specific return channel type: PHS (PIAFS2.1)

General information on the method	
int	**getARIBType**() This method acquires the ARIB-specific return channel type that the object indicates.

Details of the field

ARIB_TYPE_ETHERNET_DHCP

public static final int **ARIB_TYPE_ETHERNET_DHCP**

ARIB-specific return channel type: Ethernet (DHCP)

Reference:

Chapter 16 List of Constant Value

ARIB_TYPE_ETHERNET_FIXED_IP

public static final int **ARIB_TYPE_ETHERNET_FIXED_IP**

ARIB-specific return channel type: Ethernet (Fixed IP)

Reference:

Chapter 16 List of Constant Value

ARIB_TYPE_ETHERNET_PPPOE

public static final int **ARIB_TYPE_ETHERNET_PPPOE**

ARIB-specific return channel type: Ethernet (PPPoE)

Reference:

Chapter 16 List of Constant Value

ARIB_TYPE_MOBILE_PHONE

public static final int **ARIB_TYPE_MOBILE_PHONE**

ARIB-specific return channel type: Mobile phone (In case type information is not available.)

Reference:

Chapter 16 List of Constant Value

ARIB_TYPE_MOBILE_PHONE_CDMA_CELLUARSYSTEM
public static final int **ARIB_TYPE_MOBILE_PHONE_CDMA_CELLUARSYSTEM**
ARIB-specific return channel type: Mobile phone (CDMA CelluarSystem)
Reference:
Chapter 16 List of Constant Value

ARIB_TYPE_MOBILE_PHONE_DS_CDMA
public static final int **ARIB_TYPE_MOBILE_PHONE_DS_CDMA**
ARIB-specific return channel type: Mobile phone (DS-CDMA)
Reference:
Chapter 16 List of Constant Value

ARIB_TYPE_MOBILE_PHONE_MC_CDMA
public static final int **ARIB_TYPE_MOBILE_PHONE_MC_CDMA**
ARIB-specific return channel type: Mobile phone (MC-CDMA)
Reference:
Chapter 16 List of Constant Value

ARIB_TYPE_MOBILE_PHONE_PDC
public static final int **ARIB_TYPE_MOBILE_PHONE_PDC**
ARIB-specific return channel type: Mobile phone (PDC)
Reference:
Chapter 16 List of Constant Value

ARIB_TYPE_MOBILE_PHONE_PDCP
public static final int **ARIB_TYPE_MOBILE_PHONE_PDCP**
ARIB-specific return channel type: Mobile phone (PDC-P)
Reference:
Chapter 16 List of Constant Value

ARIB_TYPE_PHS
public static final int **ARIB_TYPE_PHS**
ARIB-specific return channel type: PHS (In case type information is not available.)
Reference:
Chapter 16 List of Constant Value

ARIB_TYPE_PHS_PIAFS20
public static final int **ARIB_TYPE_PHS_PIAFS20**
ARIB-specific return channel type: PHS (PIAFS2.0)
Reference:

Chapter 16 List of Constant Value

ARIB_TYPE_PHS_PIAFS21
public static final int **ARIB_TYPE_PHS_PIAFS21**

ARIB-specific return channel type: PHS (PIAFS2.1)

Reference:

Chapter 16 List of Constant Value

Details of the method

getARIBType
public int **getARIBType**()

This method acquires the ARIB-specific return channel type that the object indicates. This method is used to see the details of the type when TYPE_OTHER is returned by the method org.dvb.net.rc.RCInterface#getType().

Return value:

The type of return channel that this object indicates. Any constant of the ARIB-specific return channel type defined in the interface.

Annex S Application listing and launching

This Annex specifies API relevant to list acquisition and control of applications presented by AIT and conforms to GEM1.0 “Annex S: Application listing and launching”.

Annex T Permissions

This Annex specifies API relevant to conditional access and conforms to GEM1.0 “Annex T: Permissions”.

As a class that corresponds to org.dvb.net.tuning.DvbNetworkInterfaceSIUtil, jp.or.arib.tv.net.tuning.ARIBNetworkInterfaceSIUtil is specified.

T.1 Package jp.or.arib.tv.net.tuning
Extension to DAVIC tuning API

General information on the class	
ARIBNetworkInterfaceSIUtil	Each SI database corresponds to a specific network interface.

T.1.1 Explanation of Package jp.or.arib.tv.net.tuning
Extension to DAVIC tuning API

T.1.2 ARIBNetworkInterfaceSIUtil
jp.or.arib.tv.net.tuning
Class ARIBNetworkInterfaceSIUtil

java.lang.Object
  |
  +--**jp.or.arib.tv.net.tuning.ARIBNetworkInterfaceSIUtil**

public class **ARIBNetworkInterfaceSIUtil**
extends java.lang.Object

Each SI database corresponds to a specific network interface. This class provides the function to make reference to the combination with the application.

Details of the method	
static org.davic.net.tuning.NetworkInterface	**getNetworkInterface**(SIDatabase sd) This method acquires the network interface that corresponds to a specific SI database.
static SIDatabase	**getSIDatabase**(org.davic.net.tuning.NetworkInterface ni) This method acquires the SI database that corresponds to a specific network interface.

Method inherited from Class java.lang.
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Details of the method

getNetworkInterface
public static org.davic.net.tuning.NetworkInterface **getNetworkInterface**(SIDatabase sd)

This method acquires the network interface that corresponds to a specific SI database.

Parameter:

sd – SI database that needs to acquire a corresponding network interface.

Return value:

Corresponding network interface

getSIDatabase

public static SIDatabase **getSIDatabase**(org.davic.net.tuning.NetworkInterface ni)

This method acquires the SI database that corresponds to a specific network interface.

Parameter:

ni – Network interface that needs to acquire a corresponding SI database

Return value:

Corresponding SI database

Annex U Extended graphics APIs

This Annex conforms to GEM1.0 “Annex U: Extended graphics APIs”. In addition, the following APIs are added.

The following classes are added as APIs for font composition:

jp.or.arib.tv.ui.CodeSubset
jp.or.arib.tv.ui.CompositeFontFactory
jp.or.arib.tv.ui.CompositionFailedException

These APIs allow the composite font composed by combining a partial font downloaded by PFR form and a resident font of the receiver to be handled as a virtual composite font.

As for the two java.awt.Font objects for the composition, the style to be acquired by getStyle()and the size to be acquired by getSize() must coincide. In addition, there might be additional restrictions provided by the operation (e.g. Composite fonts cannot be combined each other). If the composition operation fails due to these restrictions, CompositionFaildException is generated in any case.

U.1 Package jp.or.arib.tv.ui
ARIB-specific graphics relevant function

General information on the class	
CodeSubset	Class to indicate the collection of the char value
CompositeFontFactory	Class to provide the font composition function

General information on the exception	
CompositionFailedException	The exception is generated when the font composition fails for some reason.

U.1.1 Explanation of package jp.or.arib.tv.ui
ARIB-specific graphics relevant function

U.1.2 CodeSubset
jp.or.arib.tv.ui
Class CodeSubset

java.lang.Object
  |
  +--**jp.or.arib.tv.ui.CodeSubset**

public class **CodeSubset**
extends java.lang.Object

Class to indicate the collection of the char value

General information on the method	
boolean	**contains**(char value) This method investigates whether the designated value is included in the collection.
static CodeSubset	**createCodeSubset**(char[] values) A collection of discontinuous values is generated.
static CodeSubset	**createCodeSubset**(char minValue, char maxValue) A collection of continuous values is generated.
static CodeSubset	**createCodeSubset**(int[] ranges) A collection that includes one or more Unicode character blocks is generated.

Method inherited from Class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Details of the method

contains

public boolean **contains**(char value)

This method checks whether the designated value is included in the collection.

Parameter:

value – Value to be checked

Return value:

If the designated value is included in the collection, “true” is returned.

createCodeSubset

public static CodeSubset **createCodeSubset**(char[] values)

A collection of discontinuous values is generated.

Parameter:

values – Array of code values to be included in the collection

createCodeSubset

public static CodeSubset **createCodeSubset**(char minValue,
char maxValue)

A collection of continuous values is generated. The collection to be generated includes any value x between minValue and maxValue. (minValue <= x <= maxValue)

Parameter:

minValue – Minimum code value within the range of continuous values

maxValue – Maximum code value within the range of continuous values

createCodeSubset

public static CodeSubset **createCodeSubset**(int[] ranges)

A collection that includes one or more Unicode character blocks is generated.

Parameter:

ranges – Integer array corresponding to the Unicode character block defined as a constant by org.havi.ui.HFontCapabilities.

Reference:

HFontCapabilities

U.1.3 CompositeFontFactory

jp.or.arib.tv.ui

Class CompositeFontFactory

java.lang.Object

|
+--**jp.or.arib.tv.ui.CompositeFontFactory**

public class **CompositeFontFactory**
extends java.lang.Object

Class to provide the font composition function

Reference:

org.dvb.ui.FontFactory

General information on the method	
java.awt.Font	**createCompositeFont**(java.lang.String name, java.awt.Font baseFont, java.awt.Font overridingFont) This method composes two fonts.
java.awt.Font	**createCompositeFont**(java.lang.String name, java.awt.Font baseFont, java.awt.Font overridingFont, CodeSubset mask) This method composes two fonts with a mask.
static CompositeFontFactory	**getInstance**() This method returns the unique instance of the class.

Method inherited from Class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Details of the method

createCompositeFont
public java.awt.Font **createCompositeFont**(java.lang.String name,
java.awt.Font baseFont,
java.awt.Font overridingFont)
throws ComposionFailedException

This method composes two fonts. The size and style of the target font must coincide.

Parameter:

name – Name for the font to be generated by composition.
baseFont – Font object to be the base
overridingFont – Font object for overwriting of the base font

Return value:

New Font instance generated by composition of two designated fonts

Exception:

CompositionFailedException - When the font composition failed for some reason (e.g. different font size from the designated one.)

createCompositeFont
public java.awt.Font **createCompositeFont**(java.lang.String name,
java.awt.Font baseFont,
java.awt.Font overridingFont,
CodeSubset mask)
throws CompositionFailedException

This method composes two fonts with a mask. The size and style of the target font must coincide.

Parameter:

name - Name for the font to be generated by composition.
baseFont - Font object to be the base
overridingFont - Font object for overwriting of the base font
mask – CodeSubset to be used as a mask in this composition

Return value:

New font stance generated by composition of two designated fonts

Exception:

CompositionFailedException - When the font composition fails for some reason (e.g. different font size from the designated one.)

getInstance
public static CompositeFontFactory **getInstance**()

This method returns the unique instance of the class.

Return value:

Instance of CompositeFontFactory class

U.1.4 CompositionFailedException
jp.or.arib.tv.ui
Class CompositionFailedException

java.lang.Object
|
+--java.lang.Throwable
|
+--java.lang.Exception
|
+--**jp.or.arib.tv.ui.CompositionFailedException**

Mounted interface for all:

java.io.Serializable

public class **CompositionFailedException**
extends java.lang.Exception

This exception occurs when the font composition fails for some reason (e.g. The combination of the designated fonts is inappropriate.)

Reference:

java.io.Serializable

General information on the constructor
CompositionFailedException() Default constructor
CompositionFailedException(java.lang.String reason) Generation of exception specifying the reason for the occurrence

Method inherited from Class java.lang.Throwable
fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Method inherited from Class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Details of the constructor

CompositionFailedException
public **CompositionFailedException**()

Default constructor

CompositionFailedException
public **CompositionFailedException**(java.lang.String reason)

Generation of exception specifying the reason for the occurrence

Parameter:

reason – Reason for the exception occurrence

Annex V Void

Annex W Void

Annex X Test support

The Annex specifies API for the compatibility test of receivers and conforms to GEM1.0 “Annex X: Test support”.

Annex Y Inter-application and Inter-Xlet communication API

This Annex specifies API relevant to communication among multiple applications and conforms to GEM1.0 “Annex Y: Inter-application and Inter-Xlet communication API”.

Annex Z Void

Explanatory Information

1 Relation between PMT/EIT descriptor and AIT

Fig.1 shows the relation of AIT to the PMT data compoment descriptor and EIT data contents descriptor in ARIB.

The extension items for ARIB are as follows:

- New assignment of identification value to be used in ARIB
  (e.g. data coding scheme ID, transport_id, application_id)
- For component ES that transmits ARIB-J application or for component ES that transmits AIT, selector areas are specified for data component descriptors that store additional information not to be transmitted via AIT.
- Selector areas are specified for data contents descriptors in order to store ARIB-J application details on the basis of the program event.
- In a descriptor, application_identifier_flag is set instead of entry_point_flag in order to store information specifying an application that is an entry point on PMT/EIT.

The additional definitions are for selector areas of data component descriptors as well as data contents descriptors relevant to ARIB-J and AIT transmission. The rest are supposed to conform to the AIT transmission under MHP1.0.

In addition, for the data carousel that does not transmit ARIB-J application, it is possible to refer to the contents by specifying the Locator from the ARIB-J application.

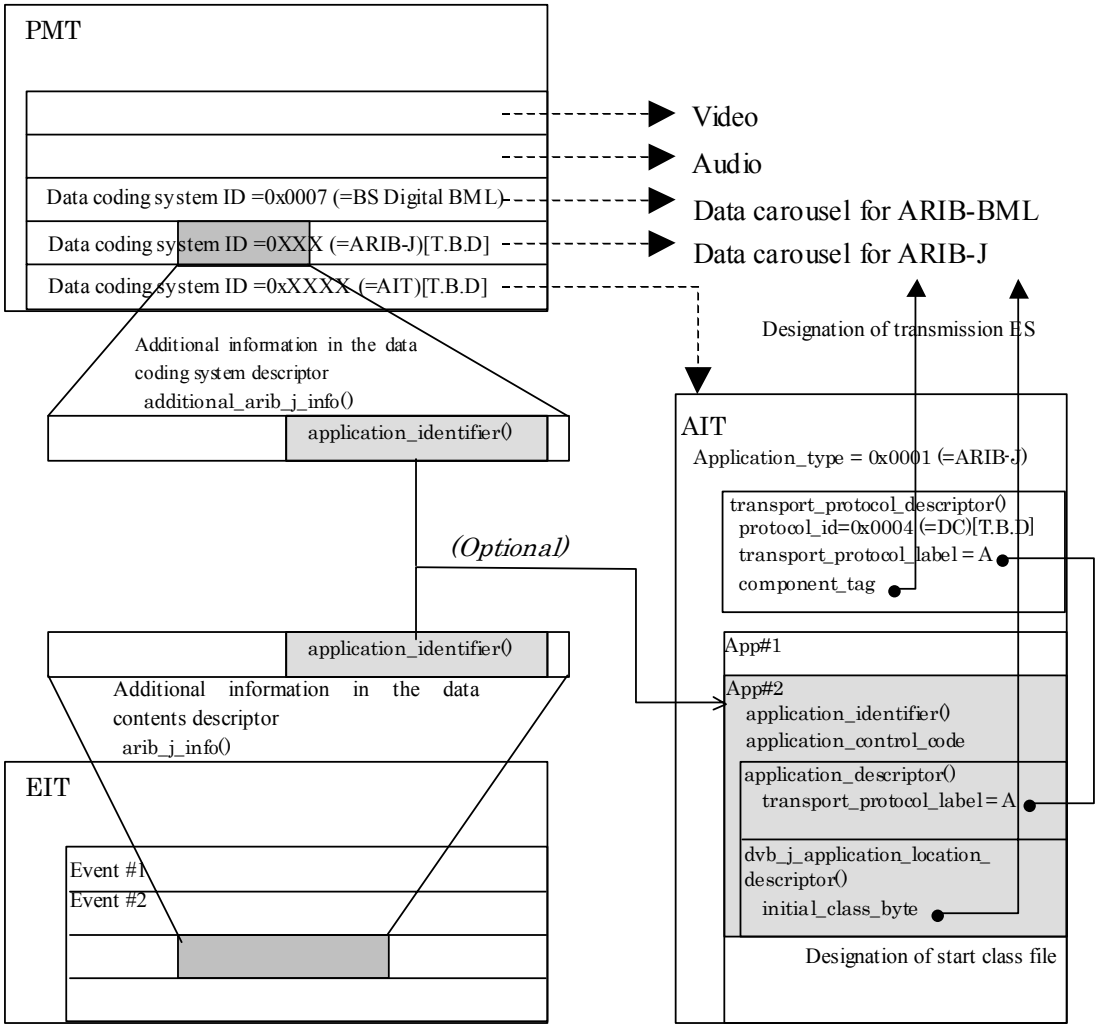


Fig. 1 Relation of the AIT transmission and data coding system descriptor in ARIB.

2 Guideline for the ARIB-AE and BML common receiver

The guideline established for a receiver that can receive both the ARIB-J application and BML document is shown below:

2.1 Mapping of plane configuration

Mapping of the memory structure in case of common use of display memory in the common receiver is shown below.

Table 1 Mapping of the display memory structure for common use

B23 Plane Configuration	B24 Plane Configuration
Background plane	Still picture plane
Video plane	Moving picture plane
——	Switching plane of still picture and moving picture
Graphic plane	Plane of character and pattern
——	Subtitle plane

Under this standard, the planes are arranged from background plane, video plane to graphic plane in the order of back to front. In ARIB STD-B24, the arrangement is defined as from moving picture plane, still picture plane, switching plane of still picture and moving picture, plane of character and pattern to subtitle plane in the order of back to front. Therefore the order should be noted when developing the common receiver of B23 and B24.

Especially, it is important to note that the positioning of the still picture display plane (background plane) and moving picture display plane (video plane) is different between B23 and B24.

2.2 Composition of still picture plane and moving picture plane

As described in MHP 1.0 “13.1 Graphics,” the area in which HVideoDevice is not set on the video plane is regarded as the image penetration area and the details of the background plane must be denoted under this standard. It is also necessary to control so that the area in which HVideoDevice is set on the moving picture plane coincides with the area of the moving picture area of the switching plane of a still picture and a moving picture when the still picture plane, moving picture plane and switching plane of a still picture and a moving picture are used.

2.3 CLUT

CLUT is described in Table 15-1 “Minimum CLUT” of Chapter 15 in this standard. However, the specification is different from the one in the operation guideline of ARIB STD-B24. The difference should be noted when a receiver that operates the BML application and the ARIB-J application simultaneously is developed.

In addition, the difference should also be noted when PNG (Portable Network Graphics)

pictures are used in both the BML application and the ARIB-J application.

Bibliography

(1) DVB Implementation Guidelines for the use of MPEG-2 Systems, Video and Audio in Satellite, Cable and Terrestrial Broadcasting Applications.

APPLICATION EXECUTION ENGINE PLATFORM FOR DIGITAL BROADCASSTING

ARIB STANDARD

ARIB STD-B23 VERSION 1.1-E1
(February 5, 2004)

This Document is based on the ARIB standard of "Application Execution Engine Platform For Digitral 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