ARIB STD-B62
Version 1.0-E1

ENGLISH TRANSLATION

MULTIMEDIA CODING SPECIFICATION FOR DIGITAL BROADCASTING (SECOND GENERATION)

ARIB STANDARD

ARIB STD-B62 Version 1.0
(Fascicle 2)

Version 1.0 July 31, 2014

Association of Radio Industries and Businesses

Foreword

The Association of Radio Industries and Businesses (ARIB) investigates and summarizes the basic technical requirements for various radio systems in the form of "ARIB Standards". These standards are developed with the participation of and through discussions amongst radio equipment manufacturers, telecommunication operators, broadcasting equipment manufacturers, broadcasters and users.

ARIB Standards include "government technical regulations" (mandatory standard) that are set for the purpose of encouraging effective use of frequency and preventing interference with other spectrum users, and "private technical standards" (voluntary standards) that are defined in order to ensure compatibility and adequate quality of radio equipment and broadcasting equipment as well as to offer greater convenience to radio equipment manufacturers, telecommunication operators, broadcasting equipment manufacturers, broadcasters and users.

This ARIB Standard is developed for "MULTIMEDIA CODING SPECIFICATION FOR DIGITAL BROADCASTING (SECOND GENERATION)". In order to ensure fairness and transparency in the defining stage, the standard was set by consensus at the ARIB Standard Assembly with the participation of both domestic and foreign interested parties from radio equipment manufacturers, telecommunication operators, broadcasting equipment manufacturers, broadcasters and users.

ARIB sincerely hopes that this ARIB Standard will be widely used by radio equipment manufacturers, telecommunication operators, broadcasting equipment manufacturers, broadcasters and users.

NOTE:
Although this ARIB Standard contains no specific reference to any Essential Industrial Property Rights relating thereto, the holders of such Essential Industrial Property Rights state to the effect that the rights listed in the Attachment 1 and 2, which are the Industrial Property Rights relating to this standard, are held by the parties also listed therein, and that to the users of this standard, in the case of Attachment 1, such holders shall not assert any rights and shall unconditionally grant a license to practice such Industrial Property Rights contained therein, and in the case of Attachment 2, the holders shall grant, under reasonable terms and conditions, a non-exclusive and non-discriminatory license to practice the Industrial Property Rights contained therein. However, this does not apply to anyone who uses this ARIB Standard and also owns and lays claim to any other Essential Industrial Property Rights of which is covered in whole or part in the contents of the provisions of this ARIB Standard.

Attachment 1 (Selection of Option 1)
(N/A)

Attachment 2 (Selection of Option 2)

Patent Holder	Name of Patent	Registration No./ Application No.	Remarks
Sony Corporation	Submitted comprehensive confirmation of patents for ARIB STD-B62 Ver1.0 [Note1]		
Sharp Corporation	Submitted comprehensive confirmation of patents for ARIB STD-B62 Ver1.0 [Note1]		

Note 1 : Valid for ARIB STD-B62 Ver1.0 (received on July 24, 2014)

TOTAL CONTENTS

VOLUME 2

Specification for Multimedia Coding Scheme

Contents

<Blank Page>

Chapter 1 General Terms

1.1 Purpose

This standard specifies the multimedia coding scheme (second generation) for the data broadcasting in digital broadcasting.

1.2 Scope

This standard applies to the multimedia coding among the data broadcasting on the advanced broadband satellite digital broadcasting.

1.3 References

1.3.1 Normative references

The following documents are those with part of their specifications quoted in this standard.

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

(2) ARIB STD-B60 "MMT-Based Media Transport Scheme in Digital Broadcasting Systems"

(3) IPTVFJ STD-0010 Version 2.0 "IPTV Forum Specification: Integrated Broadcast-Broadband System Specification"

(4) IPTVFJ STD-0011 Version 2.0 "IPTV Forum Specification: HTML5 Browser Specification"

(5) W3C Recommendation "HTML5 A vocabulary and associated APIs for HTML and XHTML" http://www.w3.org/TR/html

(6) W3C Recommendation "CSS Fonts Module Level 3" http://www.w3.org/TR/css3-fonts/

(7) W3C Recommendation "CSS Respective Specification" http://www.w3.org/Style/CSS/

(8) W3C Candidate Recommendation "W3C DOM4"

http://www.w3.org/TR/2014/CR-dom-20140508/

(9) ECMA-262(ISO/IEC 16262), ECMAScript 5th Edition

(10) IETF RFC 792 Internet Control Message Protocol

(11) IETF RFC 4443 Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification

(12) IETF RFC 1034 Domain names - concepts and facilities

(13) IETF RFC 1035 Domain names - implementation and specification

(14) IETF RFC 3986 Uniform Resource Identifier (URI): Generic Syntax

(15) ETSI TS 102 809 V1.1.1 (2010-01) DVB Signalling and carriage of interactive applications and services in Hybrid broadcast/broadband environments

1.4 Abbreviations

AIT	Application Information Table
API	Application Programming Interface
CSS	Cascading Style Sheet
DNS	Domain Name System
EIT	Event Information Table
EPG	Electronic Program Guide
FQDN	Fully Qualified Domain Name
HTML	Hypertext Markup Language
ICMP	Internet Control Message Protocol
IP	Internet Protocol
MMT	MPEG Media Transport
MMT-SI	MMT-Signaling Information
MPEG	Moving Picture Expert Group
MPEG-2 TS	MPEG-2 Transport Stream
NPT	Normal Play Time
TCP	Transmission Control Protocol
URL	Uniform Resource Locator
UTC	Coordinated Universal Time
VOD	Video On Demand
XML	eXtensible Markup Language

Chapter 2: Apply HTML5 to Television

This chapter presents how to apply W3C HTML5 Recommendation (HTML5, CSS3, Javascript function group) to TV.

For details of HTML5, refer to W3C Recommendation “HTML5 A vocabulary and associated APIs for HTML and XHTML”.

2.1 Character code

2.1.1 Character coding scheme used for HTML documents

For the character coding scheme for HTML documents, UTF-8 specified in Volume 1, Part 2, Chapter 5 shall be used.

2.1.2 External characters

For the external character coding scheme, the scheme specified in Volume 1, Part 2, 5.5 shall be used.

When external characters are used in HTML documents, the font files coded by these schemes are specified as Web Fonts.

For details of Web Fonts, refer to W3C Recommendation “CSS Fonts Module Level 3”.

2.2 Broadcast audio/video object

2.2.1 Application of an object element to broadcast audio/video

This section specifies an object element used to present a broadcast audio/video.

In this specification, a broadcast audio/video object is defined by inheriting the object element specified in W3C Recommendation HTML5. For details of the object element, refer to W3C Recommendation “HTML5 4.8.4 The object element”.

For an object element with its type attribute specified as "video/x-iptvf-broadcast", the default stream of the current channel’s broadcast service shall be presented.

Table 2-1: Attribute of the broadcast audio/video object

Attribute	Value
type	video/x-iptvf-broadcast

For the broadcast audio/video object, initial parameters to be used when the object is generated can be handed over using the param element. Even if the attributes and param element of the object are overwritten by a DOM operation using Javascript, they shall not be reflected on the receiver processing part. The relevant broadcast audio/video is controlled using functions provided by the object.

2.2.2 Broadcast audio/video object definition

The broadcast audio/video object is defined as follows.

Broadcast audio/video object definition

```
interface BroadcastVideoObjectElement : HTMLObjectElement {
  boolean enableFullscreen();
  boolean disableFullscreen();
  boolean isFullscreen();
  boolean enableAudioMute();
  boolean disableAudioMute();
  boolean isAudioMute();
  boolean setAudioSrc(DOMString url);
  DOMString getAudioSrc();
  boolean setVideoSrc(DOMString url);
  DOMString getVideoSrc();
  boolean setCaptionSrc(DOMString url);
  DOMString? getCaptionComponentURL();
  boolean isCaptionExistent(DOMString url);
  boolean setCaptionVisibility(boolean flag);
  boolean isCaptionVisible();
  void addCaptionListener(CaptionListener listener,optional DOMString url);
  void removeCaptionListener(optional CaptionListener listener);
  callback CaptionListener = void (DOMString captiondata)
};
```

`enableFullscreen`	
Description	Displays the broadcast video at the foreground in the full screen state. If `enableFullscreen()` is called in the full screen state, the full screen state shall be maintained, and true shall be returned.
Return value	true: success in making a transition to the full screen state false: failure

`disableFullscreen`	
Description	Terminates the full screen display. If `disableFullscreen()` is called in the full screen termination state, the full screen termination state shall be maintained, and true shall be returned.
Return value	true: success in terminating the full screen display false: failure

`isFullscreen`	
Description	Obtains whether or not the broadcast video display is in the full screen state.
Return value	true: the broadcast video display is in the full screen state false: Other than the above

`enableAudioMute`	
Description	Makes a transit to the mute state. If `enableAudioMute()` is called in the mute state, the mute state shall be maintained, and true shall be returned.
Return value	true: success in making a transit to the mute state false: failure

`disableAudioMute`	
Description	Terminates the mute state. If `disableAudioMute()` is called in the mute termination state, the mute termination state shall be maintained, and true shall be returned.
Return Value	true: success in terminating the mute state false: failure

`isAudioMute`	
Description	Obtains whether or not audio is in the mute state.
Return value	true: the mute state false: Other than the above

`setAudioSrc`		
Description	Selects the audio stream that is transmitted by MPEG-2 TS or MMT as a source.	
Arguments	`url`	URL that indicates the audio stream specified as a source. 4.1 and 4.2 shall be followed for format.
Return value	true: success in specifying the video stream false: failure	

`setVideoSrc`		
Description	Selects the video stream that is transmitted by MPEG-2 TS or MMT as a source.	
Arguments	`url`	URL that indicates the video stream specified as a source. 4.1 and 4.2 shall be followed for format.
Return value	true: success in specifying the video stream false: failure	

`getAudioSrc`	
Description	Obtains the stream that is selected as an audio source.
Return Value	URL that indicates the stream selected as an audio source. 4.1 and 4.2 shall be followed for format.

`getVideoSrc`	
Description	Obtains the stream that is selected as a video source.
Return value	URL that indicates the selected video stream. 4.1 and 4.2 shall be followed for format.

`setCaptionSrc`		
Description	Selects the closed caption stream that is transmitted by MPEG-2 TS or MMT as a source.	
Arguments	`url`	URL that indicates the closed caption stream to be specified. 4.5 and 4.6 shall be followed for format.
Return value	true: success in selecting the closed caption stream false: failure	

getCaptionComponentURL	
Description	Obtains URL that indicates the closed caption stream being selected currently by the receiver. Note that attention must be paid to the cases where the language identification displayed by user manipulation and others may be different from the initial setting by the param element or those specified by `setCaptionSrc()` function.
Return value	URL that indicates the closed caption stream being selected by the receiver. 4.5 and 4.6 shall be followed for format. If there is no selected closed caption, null shall be returned.

isCaptionExistent		
Description	Obtains whether or not the specified closed caption stream is currently being broadcast.	
Arguments	`url`	URL that indicates the closed caption stream to be specified. 4.5 and 4.6 shall be followed for format.
Return value	true: the specified closed caption stream is being broadcast. false: Other than the above	

setCaptionVisibility		
Description	Instructs whether or not the closed caption should be presented at the receiver.	
Arguments	`flag`	true: instruct the display of closed caption false: instruct the non-display of closed caption
Return value	true: success false: failure	

isCaptionVisible	
Description	Obtains the display state of the closed caption.
Return value	true: the closed caption is displayed false: Other than the above

addCaptionListener		
Description	Registers an event listener used to obtain the closed caption that is transmitted by MPEG-2 TS or MMT.	
Arguments	`listener`	The function that is called when receiving the closed caption stream specified by url.
	`url`	URL that indicates the closed caption to be obtained. 4.5 and 4.6 shall be followed for format. When this is omitted, the initial setting by the param element and the closed caption stream specified by `setCaptionSrc()` function are the targets to be obtained.

removeCaptionListener		
Description	Removes the event listener registered by `addCaptionListener`.	
Arguments	listener	The event listener to be removed. When this is omitted, all event listeners that are registered in the applicable broadcast audio/video object are removed.

callback CaptionListener		
Arguments	captiondata	Closed caption data. The details shall be defined in the operational rules.

A param element shall be used when specifying the operation state from the initial operation of the broadcast audio/video object. Table 2-2 shows the parameter names and values that can be specified.

Table 2-2: Broadcast audio/video object parameter list

Name	Value
`fullscreen`	`enable`: Launch in full screen mode `disable`: Disable full screen mode (default)
`video_src`	URL indicating the video stream. 4.1 and 4.2 shall be followed for format.
`audio_src`	URL indicating the audio stream. 4.1 and 4.2 shall be followed for format.
`audio_mute`	`enable`: Muto audio `disable`: Disable muting of audio (default)
`caption_src`	URL indicating the closed caption stream. 4.5 and 4.6 shall be followed for format.

2.3 VOD

To be specified in the future.

2.4 CSS

For the stylesheet to be used for HTML documents, refer to W3C Recommendation CSS Specifications.

Chapter 3: Procedure Description Language

3.1 DOM API

For DOM API to be used for HTML documents, refer to W3C DOM4.

3.2 Script description language

For the script language to be used for HTML documents, refer to ECMA-262 (ISO/IEC 16262), ECMAScript 5th Edition.

3.3 Extension function for broadcasting

3.3.1 EPG-related function

This section shall specify the interfaces relevant to program viewing schedule and video viewing schedule.

```
partial interface ReceiverDevice {
  boolean isScheduledToTune(
      ISDBResourceReference event_ref,
      optional Date startTime);
  void scheduleToTune(
      ISDBResourceReference event_ref,
      optional Date startTime);
  void unscheduleToTune(
      ISDBResourceReference event_ref);
  boolean isScheduledToRecord(
      ISDBResourceReference event_ref
      optional Date startTime);
  void scheduleToRecord(
      ISDBResourceReference event_ref,
      optional Date startTime);
  void unscheduleToRecord(
      ISDBResourceReference event_ref);
};
```

isScheduledToTune		
Description	Examines whether or not the specified program is already scheduled for viewing.	
Arguments	service_ref	Event that is surveyed
	startTime	Event start time
Return value	True if the program specified by the argument is already scheduled for viewing, false if not scheduled yet.	

scheduleToTune		
Description	Makes a viewing schedule of the specified program.	
Arguments	service_ ref	Event that is scheduled for viewing

	startTime	Event start time

unscheduleToTune		
Description	Cancels a viewing schedule of the specified program.	
Arguments	service_ref	Event whose viewing schedule is cancelled

isScheduledToRecord		
Description	Examines whether or not the specified program is already scheduled for video recording.	
Arguments	service_ref	Event that is surveyed
	startTime	Event start time
Return value	True if the program specified by the argument is already scheduled for video recording, false if not scheduled yet.	

scheduleToRecord		
Description	Makes a video recording schedule of the specified program.	
Arguments	service_ref	Event that is scheduled for video recording
	startTime	Event start time

unscheduleToRecord		
Description	Cancels a video recording schedule of the specified program.	
Arguments	service_ref	Event whose viewing schedule is cancelled

3.3.2 Series schedule function

This section shall specify the interfaces relevant to the program viewing schedule and video viewing schedule of the series specified by a series descriptor.

```
partial interface ReceiverDevice {
  boolean isScheduledToTuneSeries(
      SeriesReference series_ref,
      Date expire_date);
  void scheduleToTuneSeries(
      CurrentEventInformation service_ref,
      Date expire_date);
  void unscheduleToTuneSeries(
      CurrentEventInformation service_ref
      Date expire_date);
  boolean isScheduledToRecordSeries(
      CurrentEventInformation service_ref,
      Date expire_date);
  void scheduleToRecordSeries(
      CurrentEventInformation service_ref,
      Date expire_date);
  void unscheduleToRecordSeries(
      CurrentEventInformation service_ref,
      Date expire_date);
};
```

```
dictionary SeriesReference : ISDBResourceReference {
  attribute unsigned short series_id;
};
```

isScheduledToTuneSeries		
Description	Examines whether or not a viewing schedule is already made for the specified series.	
Arguments	series_ref	Series that is surveyed
	expire_date	Expiration date of series
Return value	True when a series viewing schedule was already made for the program specified by the argument, false if not scheduled yet.	

scheduleToTuneSeries		
Description	Makes a viewing schedule of the specified series.	
Arguments	series_ref	Series that is scheduled for viewing
	expire_date	Expiration date of series

unscheduleToTuneSeries		
Description	Cancels a viewing schedule of the specified series.	
Arguments	series_ref	Series whose viewing schedule is cancelled
	expire_date	Expiration date of series

isScheduledToRecordSeries		
Description	Examines whether or not a video recording schedule was already made for the specified series.	
Arguments	series_ref	Series that is surveyed
	expire_date	Expiration date of series
Return value	True if a series video recording schedule is already made for the program specified by the argument, false if not scheduled yet.	

scheduleToRecordSeries		
Description	Makes a video recording schedule of the specified program.	
Arguments	series_ref	Series that is scheduled for video recording.
	expire_date	Expiration date of series

unscheduleToRecordSeries		
Description	Cancels a video recording schedule of the specified series.	
Arguments	series_ref	Series whose video recording schedule is cancelled.
	expire_date	Expiration date of series

3.3.3 Closed caption display control function

It is assumed that HTML application access the closed caption data that are transmitted by the method specified in ARIB STD-B24 Volume 1 Chapter 9 or specified in ARIB STD-B60 Chapter 9. The closed captions to be transmitted are presented by the receiver function.

In the broadcast audio/video object shown in 2.2, the initial operating parameters of closed captions referred to shall be specified using a param element. For details of parameters, see the specification for broadcasting object shown in 2.2.2. Regarding the arrangement of the broadcast audio/video object element and param element, however, the application engine shall not conduct the presentation control over closed captions but shall conduct the control over closed captions by using API that gives an instruction for the presentation of closed captions indicated in 2.2.2.

3.3.4 Root certificate-related function in encrypted communication

To be specified in the future.

3.3.5 Local memory domain utilized from application

The access to the persistent memory domain in shared receivers shall conform to the IPTV Forum Specification: IPTVFJ STD-0011 3.1.13.2.1. Other persistent memory domains, and the functions equivalent to Ureg, Greg that are specified in ARIB STD-B24 Volume 2, 7.6.15 and 7.6.16 shall be specified in the next version.

3.3.6 Communication function with its TCP/IP connection assumed

This section specifies the interface for confirmation of the connecting state to an IP network.

```
partial interface ReceiverDevice {
    boolean confirmIPNetwork(
        DOMString destination,
        unsigned short confirmType,
        optional unsigned short timeout);
};
```

confirmIPNetwork		
Description	Confirms the connecting state to an IP network.	
Arguments	destination	IP address that is used for a communication destination to confirm the communication connecting state, or the character string that is used to specify FQDN of the host name.
	confirmType	One of the following two values shall be specified as a means to confirm the connecting state to an IP network. 0: trial run for the name solution in obedience to DNS setting 1: confirm whether or not the Echo Reply message of ICMP Echo message is received.
	timeout	Specifies the time for waiting for a response (unit in milliseconds).
Return Value	true: success false: failure	

3.3.7 Application manager object

This section specifies the application manager object that provides the interface for application execution control. The application manager object is provided as the property of the Navigator object. The constructor of the application manager object is not provided.

3.3.7.1 Interface definition

```
[NoInterfaceObject]
interface NavigatorApplicationManager {
    readonly attribute ApplicationManager applicationManager;
};

Navigator implements NavigatorApplicationManager;

[NoInterfaceObject]
interface ApplicationManager {
    Application? getOwnerApplication(optional Document document);
};
```

3.3.7.2 Method

getOwnerApplication		
Description	Returns the application to which the document indicated by the argument *document* belongs.	
Arguments	document	Object identifying the HTML document that wants to obtain information about the application to which it belongs. When this argument is omitted, it is deemed that the object that identifies the HTML document that has executed this method is specified.
Return value	Application object that identifies the application whose information has been obtained. Null if no applicable application exists.	

3.3.8 Application object

This section specifies the application object that is an object identifying an application. The application object is returned by the method of application manager object getOwnerApplication that was specified in the previous section. The constructor of the application object is not provided.

3.3.8.1 Interface definition

```
[NoInterfaceObject]
interface Application {
    readonly attribute DOMString type;
    readonly attribute unsigned long long organization_id;
    readonly attribute unsigned long long application_id;
    readonly attribute DOMString control_code;
    void replaceApplication(
        unsigned long long organization_id,
        unsigned long long application_id,
        DOMString? ait_url);
    void destroyApplication();
    void exitFromManagedState(DOMString url);
    ApplicationInformationTable getOwnerAIT();
    ApplicationBoundaryAndPermissionDescriptor?
        getApplicationBoundaryAndPermissionDescriptor();
};

[NoInterfaceObject]
interface ApplicationBoundaryAndPermissionDescriptor {
    void addPermissionManagedArea(
        sequence<unsigned short>? permission,
        sequence<DOMString>? urls);
};
```

3.3.8.2 Property

type	
Description	Application format for the given application. The value of this property is the character string that is specified in 5.3.3 as the value of an applicationType element.

organization_id	
Description	System identification for the given application. For the system identification, refer to 5.2.2.

application_id	
Description	Application identification for the given application. For the application identification, refer to 5.2.2.

control_code	
Description	Application control code for the given application. The value of this property is any one of the character strings that are specified in 5.3.3 as the values of a controlCode element.

3.3.8.3 Method

replaceApplication		
Description	Terminates the application that has executed this method, and launches the application specified by the argument.	
Arguments	organization_id	Organization ID of the application to be launched.
	application_id	Application ID of the application to be launched.
	ait_uri	URL identifying the location of the AIT or null. The operation when null is specified is defined in the operational rules.

destroyApplication	
Description	Terminates the application that has executed this method. The behavior of the application engine after the termination of application follows IPTV Forum Specification: IPTVFJ STD-0010 7.4.3.

exitFromManagedState		
Description	Makes a transition to a general application. Terminates the application that has executed this method, and makes a transition to the document specified by the argument in an unmanaged state.	
Arguments	url	Entry URL of the general application to which the transition is to be made.

`getOwnerAIT`	
Description	Obtains the application information table (AIT) that controls the given application that has executed this method.
Return value	Object identifying the obtained AIT

`getApplicationBoundaryAndPermissionDescriptor`	
Description	Obtain the object that identifies the application boundary authority setting descriptor that was arranged in the given application. For the application boundary authority setting descriptor, refer to ARIB STD-B24 or ARIB STD-B60.
Return value	Object identifying the obtained application boundary authority setting descriptor, or null when this descriptor is not arranged in the given application.

`addPermissionManagedArea`		
Description	Adds the access authority management area of the application. The execution of this method is equivalent to adding one loop of the application boundary authority setting descriptor. The access authority management area added by this method is initialized when the AIT for which the given application is under its control is updated.	
Arguments	`permission`	Array whose elements are a bit map representing the access authority of the application in the access authority management area to be added, or null which signifies "maximum permission". If an array with no elements is specified, it is deemed that null is specified.
	`urls`	Area to be added to the access authority management area, or an array whose elements are URL strings, or null, which signifies "all other URLs".

3.3.9 ApplicationInformationTable object

This section specifies the ApplicationInformationTable object that is the object identifying the application information table (AIT). The ApplicationInformationTable object is returned by the getOwnerAIT method of the application object. The constructor of the ApplicationInformationTable object is not provided.

3.3.9.1 Interface definition

```
[NoInterfaceObject]
interface ApplicationInformationTable {
    sequence<Application> getApplications();
};
```

3.3.9.2 Method

`getApplications`	
Description	Obtains all the applications that are arranged in the given AIT.
Return value	Array where the application objects identifying applications that are arranged in the given AIT are all stored in the same order as the appearance order in the given AIT.

3.3.10 Capabilities object

This section specifies the capabilities object that provides information about the scope of functions provided by the application engine and the receiver platform. The capabilities object is provided as the capabilities property of the Navigator object. The constructor of the capabilities object is not provided.

```
[NoInterfaceObject]
interface NavigatorCapabilities {
    readonly attribute Capabilities capabilities;
};

Navigator implements NavigatorCapabilities;

[NoInterfaceObject]
interface Capabilities {
    boolean hasCapability(DOMString query, DOMString ... params);
};
```

hasCapability		
Description	Obtains information about whether the application engine or the receiver platform has the functions identified by the argument.	
Arguments	`query`	Character string identifying the function that is subject to query. The strings that can be specified and their meanings shall be defined in the operational rules.
	`params`	Character string identifying supplementary information about the query according to the character string specified in the argument *query*. The character string that can be specified and its meanings shall be defined in the operational rules.
Return value	True if the function specified by the argument is present, false if not.	

3.3.11 ReceiverDevice object

This section specifies the ReceiverDevice object that provides the interface for accessing the function provided by the device in which the application engine is operating, or for accessing the information managed by this device. The ReceiverDevice object is provided as the receiverDevice property of the Navigator object. The constructor of the ReceiverDevice object is not provided.

There may be some restrictions on how to call a method of the ReceiverDevice object depending on the way the receiver is implemented (such as the maximum number of parallel executions, combinations of methods that cannot be executed simultaneously). If restrictions are required, their details shall be defined in the operational rules.

```
[NoInterfaceObject]
interface NavigatorReceiverDevice {
    readonly attribute ReceiverDevice receiverDevice;
};
```

```
Navigator implements NavigatorReceiverDevice;

[NoInterfaceObject]
interface ReceiverDevice {
};
```

3.3.11.1 Obtaining the receiver-unique identifier

```
partial interface ReceiverDevice {
    void getDeviceIdentifier(long type,
            DeviceIdentifierCallback resultCallback);
};
callback DeviceIdentifierCallback = void (DOMString? identifier);
```

getDeviceIdentifier		
Description	Returns a receiver-unique identifier of the type specified by the argument `type`. The details of the values that can be specified in the argument `type`, and the character string to be returned shall be specified for each service operation. Since the identifier returned by this method can be associated with personal information of the receiver user, the application creator using this method should pay due consideration to handling the identifier returned by this method.	
Arguments	`type`	Value indicating the type of the receiver-unique identifier to be obtained.
	`resultCallback`	Function to be called when the processing is completed.

callback DeviceIdentifierCallback		
Arguments	`identifier`	Obtained receiver-unique identifier, or null, which indicates that the attempt to obtain the identifier has failed.

3.3.11.2 Obtaining information about product

The getSystemInformation method of the ReceiverDevice object is specified as an Interface through which the application obtains the information about the receiver as a product. While it is assumed that information that can be obtained with this method includes the receiver manufacturer and the software version of the application engine, the scope of this information shall be specified in the operational rules.

```
partial interface ReceiverDevice {
    object getSystemInformation(sequence<DOMString>? query);
};
```

getSystemInformation		
Description	Obtains information about the application engine or the receiver.	
Arguments	`query`	Character string array identifying the name of the item to be queried. The item name that can be specified shall be defined in the operational rules. If this is omitted, it is assumed that an instruction is given to obtain the default item specified in the operational rules.

Return value	Object that stores information about the item requested in the argument. The property name of this object is the item name, and the property value of the object is the value of that item. If no information can be returned about any specified items, the object without property shall be returned.

3.3.11.3 Channel selection

```
partial interface ReceiverDevice {
    void tuneTo(ISDBResourceReference service_ref,
            TuneToResultCallback? resultCallback,
            optional TuneToOptions options);
};
callback TuneToResultCallback = void (
    ISDBResourceReference? service_ref);

dictionary TuneToOptions {
    attribute boolean unbound = false;
};
```

`tuneTo`			
Description	Changes the service being received. Even if the specified service is the same as the service being received (except for a case where "true" is set in the argument *unbound*, which makes these regarded as the same), the channel selection operation is executed. If there is any broadcast video and/or audio present, they continue to be broadcast without interruption. Except for the case where true is specified in argument *unbound* and the application continues to be executed, the application engine may terminate before this function sends back a return value.		
Arguments	`service_ref`	Object identifying the service to be changed	
	`resultCallback`	Function to be called when the processing is completed. Null if this is not required.	
	`options`		
		`unbound`	If the service identified by `service_ref` is not the currently received service, and if the property is true, the service identified by service_ref is regarded as the same as the currently received service and continued execution of the application is attempted. After the service is changed, the operation continues if signals that allow the application to continue to be executed are received in this service. Otherwise, the operation is terminated.

`callback TuneToResultCallback`		
Arguments	`service_ref`	Object identifying the service that has become the current service after channel selection. Or null, which indicates a failure in channel selection.

3.3.11.4 Obtaining information about the event information table (EIT) [current/following]

```
partial interface ReceiverDevice {
    void getCurrentEventInformation(
            CurrentEventInformationCallback resultCallback);
};
callback CurrentEventInformationCallback = void (CurrentEventInformation
info);

dictionary CurrentEventInformation : ISDBResourceReference {
    attribute Date start_time;
    attribute long long duration;
    attribute DOMString name;
    attribute DOMString desc;
    attribute unsigned short f_event_id;
    attribute Date f_start_time;
    attribute long long f_duration;
    attribute DOMString f_name;
    attribute DOMString f_desc;
};
```

getCurrentEventInformation		
Description	Returns information about the EIT [current/following].	
Arguments	resultCallback	Function to be called when the processing is completed.

callback CurrentEventInformationCallback			
Arguments	info	Information about the current event obtained as a result of the processing	
		start_time	Value representing start_time in the event information section of EIT [current] in Date format
		duration	Value representing duration in the event information section of the EIT [current] in milliseconds
		name	Character string representing the value of program name (event_name_char) in the short event descriptor of the EIT [current]
		desc	Character string representing the program description (text_char) in the short event descriptor of the EIT [current]
		f_event_id	Value representing event_id in the event information section of the EIT [following]
		f_start_time	Value representing start_time in the event information section of the EIT [following] in Date format
		f_duration	Value representing duration in the event information section of the EIT [following] in milliseconds
		f_name	Character string representing the value of the program name (event_name_char) in the short event descriptor of the EIT [following]

		f_desc	Character string representing the value of program description (text_char) in the short event descriptor of the EIT [following]

3.3.12 Stream event target object

This section specifies the stream event target object that is the interface with which the application to use events is delivered as broadcast signals.

```
[NoInterfaceObject]
interface StreamEventTarget {
};

partial interface ReceiverDevice {
    readonly attribute StreamEventTarget streamEvent;
};
```

3.3.12.1 Reception of generic event messages

This section specifies the interface with which the application uses generic event messages that are specified in ARIB STD-B24 or MH generic event messages that are specified in ARIB STD-B60.

```
partial interface StreamEventTarget {
    void addGeneralEventMessageListener(
        GeneralEventMessageListenerParams param,
        GeneralEventMessageListener listener);
    void removeGeneralEventMessageListener(
        GeneralEventMessageListenerParams param,
        optional GeneralEventMessageListener listener);
};
callback GeneralEventMessageListener = void (GeneralEventMessage msg);

dictionary GeneralEventMessageListenerParams {
    attribute ISDBResourceReference component_tag;
    attribute unsigned short message_group_id;
    attribute octet message_id;
    attribute octet message_version;
};

dictionary GeneralEventMessage {
    attribute ISDBResourceReference component_tag;
    attribute unsigned short message_group_id;
    attribute octet message_id;
    attribute octet message_version;
    attribute DOMString? private_data_byte;
};
```

`addGeneralEventMessageListener`			
Description	Registers the event listener of a generic event message.		
Arguments	`param`	`component_tag`	Object identifying the component to be monitored
		`message_group_id`	Message group identifier of the applicable event message
		`message_id`	Message identifier of the applicable event message
		`message_version`	Message version of the applicable event message
	`listener`	Function that should be called when an event message that satisfies the requirements specified by the argument *`param`* has been received, and the ignition time described in the applicable event message has arrived. If event messages with the same message identifier are received several times, this function is executed only when the message version of the given event message received at the second or later times is different from that of the previously received event message.	

`removeGeneralEventMessageListener`			
Description	Removes the event listener of a generic event message. The event listeners to be removed are those specified by the argument *`param`*, and at the same time those whose function is the function specified by the argument `listener`.		
Arguments	`param`	`component_tag`	Object identifying the component to be monitored by the event listener to be removed
		`message_group_id`	Message group identifier of the event message targeted by the event listener to be removed
		`message_id`	Message identifier of the event message targeted by the event listener to be removed
		`message_version`	Message version of the event message targeted by the event listener that is to be removed
	`listener`	Function of the event listener to be removed. When this is omitted, it is deemed that an instruction is given to remove all event listeners targeting the event message specified by the argument *`param`* irrespective of the event listener function.	

`callback GeneralEventMessageListener`			
Arguments	`msg`	Information about the ignited event message	
		`component_tag`	Component to which the applicable event message was sent
		`message_group_id`	Message group identifier of the applicable event message
		`message_id`	Message identifier of the applicable event message
		`message_version`	Message version of the applicable event message

		private_data_byte	Private data byte of the applicable event message

Handling of each property in the argument *param* of addGeneralEventMessageListener method

message_group_id	The application may specify or omit this property. The operation when this property is already omitted is specified in the operational rules.
message_id	The application may specify or omit this property. When this property is omitted, the application engine assumes that any event message has the appropriate message identifier.
message_version	When the application has omitted message_id property, the application engine ignores this property. Otherwise, the application may either specify or omit this property. When this property is omitted, the application engine assumes that any event message has the appropriate message identifier.

Handling of each property in the argument *param* of removeGeneralEventMessageListener method

message_group_id	The application may specify or omit this property. The value that can be specified is "1". How the application engine handles any value other than "1" is to be specified later. If this property is omitted, the application engine assumes that "1" is specified.
message_id	The application may specify or omit this property. When this property is omitted, the application engine does not include the message identifier, which is a target of the event listener, in its requirements at the time of deciding the event listener to be removed.
message_version	The application may specify or omit this property. When this property is omitted, the application engine does not include the message version, which is a target of event listener, in its requirements at the time of deciding the event listener to be removed.

Handling of each property in the argument *msg* of GeneralEventMessageListener

message_group_id	The application engine must set up all of these properties without fail.
message_id	
message_version	
private_data_byte	

3.3.12.2 Reception of timer event based on NPT

This section specifies the interface with which the application to use the timer event based on NPT is specified in ARIB STD-B24.

```
partial interface StreamEventTarget {
    void addNPTReferenceMessageListener(
        ISDBResourceReference component_tag,
        NPTReferenceMessageListener listener);
    void removeNPTReferenceMessageListener(
        ISDBResourceReference component_tag,
        optional NPTReferenceMessageListener listener);
};
callback NPTReferenceMessageListener = void (ISDBResourceReference
component_tag);

partial interface StreamEventTarget {
    unsigned long setAlarmByNPT(
        ISDBResourceReference component_tag,
        unsigned long long npt_value,
        NPTAlarmHandler handler);
    void unsetAlarmByNPT(
        unsigned long handle);
};
callback NPTAlarmHandler = void (
    ISDBResourceReference component_tag,
    unsigned long long npt_value);

partial interface StreamEventTarget {
    unsigned long long getNPT(ISDBResourceReferece component_tag);
};
```

addNPTReferenceMessageListener		
Description	Registers an event listener that is executed when the function for using NPT becomes usable	
Arguments	component_tag	Object identifying the component to be monitored.
	listener	Function that is called when an NPT reference descriptor has been received by the receiver, and has associated NPT with UTC, and a function that uses NTP becomes usable. If the application is in the relevant state at the time when this method is executed, the function specified here is called immediately without waiting for reception of the next NPT reference descriptor. After this function has been executed while the application is already in the relevant state, this function is not called again, even if a new NPT reference descriptor is received.

removeNPTReferenceMessageListener		
Description	Removes the event listeners registered in addNPTReferenceMessageListener	
Arguments	component_tag	Object identifying the component to be monitored by the event listener to be removed
	listener	Event listener to be removed. When this is omitted, it is assumed that an instruction is given to remove all event listeners where the components shown in argument component_tag are registered to be monitored.

setAlarmByNPT		
Description	Registers the processing to be executed at the specified NPT time.	
Arguments	component_tag	Object identifying the component to be registered.
	npt_value	NPT value identifying the time when the function specified by the argument *handler* should be executed.
	handler	Function to be executed when the NPT time specified in argument npt_value has arrived in the component specified in component_tag. If the given NPT time has already elapsed when this method is executed, this function is executed immediately.
Return value	Handle identifying the registered processing.	

unsetAlarmByNPT		
Description	Cancels the registration made using setAlarmByNPT.	
Arguments	handle	Handle identifying what is cancelled. The value returned by setAlarmByNPT that executes the registration to be cancelled is specified.

callback NPTAlarmHandler		
Arguments	component_tag	Component specified in the argument component_tag at the time when setAlarmByNPT is executed.
	npt_value	NPT value specified in the argument npt_value at the time when setAlarmByNPT is executed.

getNPT		
Description	Obtains the current NPT value.	
Arguments	component_tag	Object identifying the component to be processed.
Return value	Obtained NPT value.	

3.3.12.3 Reception of timer event based on UTC-NPT

This section specifies the interface with which the application to use the timer event based on UTC-NPT is specified in ARIB STD-B62.

```
partial interface StreamEventTarget {
    void addUTCNPTReferenceMessageListener(
        UTCNPTReferenceMessageListenerParams param,
        UTCNPTReferenceMessageListener listener);
    void removeUTCNPTReferenceMessageListener(
        UTCNPTReferenceMessageListenerParams param,
        optional UTCNPTReferenceMessageListener listener);
};
callback UTCNPTReferenceMessageListener = void (UTCNPTNotification);

partial interface StreamEventTarget {
    unsigned long setAlarmByUTCNPT(
        UTCNPTAlarmParams param,
        UTCNPTAlarmHandler handler);
    void unsetAlarmByUTCNPT(
        unsigned long handle);
};
callback UTCNPTAlarmHandler = void (UTCNPTNotification msg);

partial interface StreamEventTarget {
    unsigned long long getUTCNPT(UTCNPTMessageListenerParams param);
};

dictionary UTCNPTReferenceMessageListenerParams {
    /* TBD */
};

dictionary UTCNPTNotification {
    unsigned long long utcnpt_value;
};
```

addUTCNPTReferenceMessageListener		
Description	Registers an event listener that is executed when the function for using UTC-NPT becomes usable.	
Arguments	param	This is the argument for compatibility, and the application engine must ignore the value safely even if any value is specified for this argument.
	listener	Function that is called when a UTC-NPT reference descriptor has been received by the receiver, and has associated UTC-NPT with UTC, and a function that uses UTC-NPT becomes usable. If the application is in the relevant state at the time when this method is executed, the function specified here is called immediately without waiting for reception of the next UTC-NPT reference descriptor. After this function has been executed while the application is already in the relevant state, this function is not called again, even if new UTC-NPT reference descriptor is received.

removeUTCNPTReferenceMessageListener		
Description	Removes the event listener registered in addUTCNPTReferenceMessageListener.	
Arguments	param	This is the argument for compatibility, and the application engine should ignore the value safely even if any value is specified for this argument.
	listener	Event listener to be removed. When this is omitted, it is assumed that an instruction is given to remove all event listeners that are registered.

setAlarmByUTCNPT			
Description	Registers the processing that should be executed at the specified UTC-NPT time.		
Arguments	param	utcnpt_value	UTC-NPT value identifying the time when the function specified by the argument *handler* should be executed.
	handler		Function to be executed when the UTC-NPT time specified by the argument npt_value has arrived in the component specified in component_tag. If the given UTC-NPT time has already elapsed when this method is executed, this function is executed immediately.
Return value	Handle identifying the registered processing.		

unsetAlarmByUTCNPT		
Description	Cancels the registration conducted by setAlarmByUTCNPT.	
Arguments	handle	Handles identifying the target to be cancelled. setAlarmByUTCNPT that conducted the registration to be cancelled specifies the returned value.

callback UTCNPTAlarmHandler			
Arguments	msg	utcnpt_value	UTC-NPT value that was specified when setAlarmByUTCNPT was executed.

getUTCNPT		
Description	Obtains present UTC-NPT value.	
Arguments	param	This is the argument for compatibility, and the application engine should ignore the value safely even if any value is specified for this argument.
Return value	Obtained UTC-NPTvalue.	

3.3.13 Cache control of data resource

To be specified in the next version.

3.4 Security model

3.4.1 Requested function

From the standpoint of protecting broadcasting contents and viewers, there is a need to establish a mechanism capable of managing applications and controlling their operating range, and a mechanism capable of excluding noncompliant receivers. In order to realize these mechanisms, the security function requested for the management of applications in receivers conforms to IPTV Forum Specification: IPTVFJ STD-0010 6.3.1. However, this standard does not apply to the functions relating to enforcement and revocation.

3.4.2 Access control

This shall conform to IPTV Forum Specification: IPTVFJ STD-0010 6.3.2.

3.4.3 Application boundary and broadcasting resource access

This shall conform to IPTV Forum Specification: IPTVFJ STD-0010 6.1.2.

Chapter 4: Namespace

4.1 Identification of resource obtained by MPEG-2 TS transmission

When the audio/video streams transmitted by MPEG-2 TS are specified, they are identified uniquely by the following names. For details of each parameter, refer to ARIB STD-B24 Volume 2 "9.2.3 Reference of AV streams and subtitle component". Furthermore, for their shortened forms, refer to ARIB STD-B24 Volume 2 "9.2.3.1 Abbreviated AV stream names". However, the provisions for data carousel transmission of AV streams are excluded. In addition, for the identification of broadcasting services currently under channel selection, refer to ARIB STD-B24 Volume 2 "9.2.5.1 Identification of currently selected broadcasting service on receiver".

arib://*<original_network_id>.<transport_stream_id>.<service_id>[;<content_id>]*
[.<event_id>]/<component_tag>[;<channel_id>]

4.1.1 Abbreviated AV stream names transmitted by MPEG-2 TS

When an AV stream is specified in the following format:

/<component_tag>[;<channel_id>]

It is interpreted as:

arib://....[<event_id>]/<component_tag>[;<channel_id>]

4.2 Identification of audio/video by MMT

When the audio/video streams transmitted by MMT are specified, they are identified uniquely by the following names.

arib-tlv://*<original_network_id>.<tlv_stream_id>.<service_id>[.<event_id>]/<component_tag>*

For the formats for parameters with the exception of <*tlv_stream_id* >, refer to ARIB STD-B24 Volume 2 "9.2.3 References of AV streams and subtitle component". <*tlv_stream_id*> is specified in hexadecimal strings as in the case of <*transport_stream_id*>. The characters (strings) indicating that these are written in hexadecimal notation such as "0x" at the top and "h" at the end of character strings are not attached, but 0 is attached at the top if required so that the fixed-length string of length 4 is created.

4.2.1 Identification of broadcasting services currently under channel selection for the Av stream transmitted by MMT

If the special value"-1" is set respectively to <*original_network_id*>, <*tlv_stream_id*>, <*service_id*>, this is interpreted as the broadcasting service currently under channel selection is being specified by receivers.

For example, if

arib-tlv://-1.-1.-1/-1

is specified as the argument of setVideoSrc function, this is interpreted as the default video stream of the broadcasting service currently under channel selection is being specified by receivers.

4.3 Identification of data resource acquisition by MMT

For the reference of data resources transmitted by MMT, they are identified uniquely by the following name.

http://[<*hostname*>]/<*directory*>/.../<*filename*> or

https://[<*hostname*>]/<*directory*>/.../<*filename*>

<*hostname*> indicates the host name of the server that can be omitted and where the resource is in existence. <*directory*> indicates the name of the directory from the root directory of the server specified by the host name. <*filename*> indicates the name of the file attached to the resource. The maximum value of the layered structure of the directory name and the maximum length of the character string of each parameter shall be defined separately in the operational rules.

No distinction is made between upper- and lower-case alphabetic characters in the characters specified in<*filename*>, but for example a judgment is made that "abc" and "ABc" are originated from the same resource.

The characters that can be used in <*hostname*>, <*directory*>, <*filename*> are as follows.

filename = startChar *echar

echar = startChar | "-" | "." | "!" | "~" | """" |

"(" | ")" | ";" | "/" | "@" | "=" | "+" | "$" | ","

startChar = lowalpha | upalpha | digit | "_"

lowalpha = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" |

"j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" |

"s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"

upalpha = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" |

"J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" |

"S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"

digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |"8" | "9"

4.4 Scheme of resource obtained from IP transmission

Resource acquisition that explicitly specified IP transmission is uniquely identified by the URL that is specified in IETF RFC3986. The specified scheme is as follows.

http

https

The maximum length of the URL shall be defined separately in the operational rules.

4.5 Identification of closed caption by MPEG-2 TS transmission

The closed caption component of the broadcasting wave transmitted by MPEG-2 TS is uniquely identified by the following name. A shortened form is not used. For details of each parameter with the exception of <*module_id*>, refer to ARIB STD-B24 Volume 2 "9.2.3 Reference of AV streams and subtitle component".

arib://<*original_network_id*>.<*transport_stream_id*>.<*service_id*>[;<content_id>]
[.<event_id>]/<component_tag>[;<module_id>]

When the component tag was specified as -1, the closed caption component currently selected was regarded as specified. <*module_id*> is used to uniquely identify languages. <*module_id*> employs hexadecimal strings. However, the characters (strings) indicating that these are written in hexadecimal notation such as "0x" at the top and "h" at the end of the character strings are not attached, but 0 is attached at the top if required so that the fixed-length string of length 4 is created. For instance, if the module identification is 0x0001, 0001 is specified in <*module_id*>.

4.6 Identification of closed caption by MMT

The closed caption component transmitted by MMT is uniquely identified by the following name.

arib-tlv://<*original_network_id*>.<*tlv_stream_id*>.<*service_id*>[.<*event_id*>]/<*component_tag*>

For details of each parameter, 4.2 shall be followed.

4.7 Identification of receiver built-in sound

Receiver built-in sound is identified by the character string:

romsound://<*sound_id*>

Here <*sound_id*> is the value of the identifier expressed by the decimal string that is used to identify the types of built-in sounds defined separately in the operational rules.

Chapter 5: Application Control Information

5.1 Application control information of section format

The application control information of section format that is used in the broadcasting system using MPEG-2 TS shall be specified in ARIB STD-B24 Volume 4 Chapter 5.

5.1.1 Application type

0x0011 shall be used as the value showing the application type (ARIB-HTML5 Application).

5.2 Application control information of MMT-SI format

The application control information of MMT-SI format that is used in the broadcasting system using MMT shall be specified in ARIB STD-B60 as MH-Application Information Table (MH-AIT).

5.2.1 Application type

0x0011 shall be used as the value showing the application type (ARIB-HTML5 Application).

5.2.2 Identification of application

The application is uniquely identified by the application identifier shown in Table 5-1. This identifier is composed of a structure with 6 bytes (48 bits) in length, and is stored in MH-AIT.

Table 5-1: Structure of Application Identifier

Data Structure	Number of Bits	Bit String Notation
application_identifier(){		
organization_id	16	uimsbf
application_id	32	uimsbf
}		

Meaning:

organization_id (system identification): indicates the system that prepared the application. This identification specifies the number uniquely given.

application_id (application identification): indicates the number that identifies the application. The number is uniquely given in system identification.

5.2.3 Application control code

The application control code specified in ARIB STD-B60 shall be used.

5.3 Application control information of XML format

The application control information of XML format shall conform to ETSI TS 102 809 V1.1.1 (2010-01) DVB Signalling and carriage of interactive applications and services in

Hybrid broadcast/broadband environments “5.4 XML-based syntax” with the extension attached.

Figure 5-1 shows the superior structure of AIT of XML format.

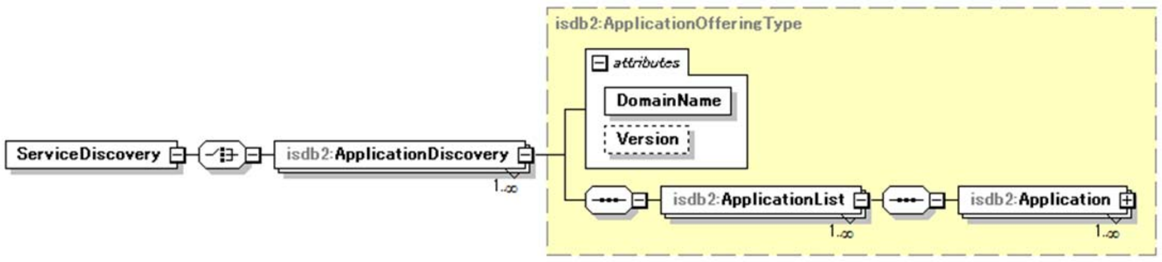

Fig.5-1: Superior structure of AIT of XML format

A ServiceDiscovery element is present at the top, an ApplicationDiscovery element down below, and an ApplicationList element further down below that. Each element up to these elements conforms basically to ETSI TS 102 809 V1.1.1 (2010-01) DVB Signalling and carriage of interactive applications and services in Hybrid broadcast/broadband environments “5.4 XML-based syntax”, but the Application element down below, and the elements further down below that shall be additionally specified as follows. As the namespace, each superior element shall also be specified as isdb2:ApplicationDiscovery, isdb2:ApplicationList.

5.3.1 Application element

As an Application element, the following XML schema shall be additionally applied. This element responds to part of the structure of AIT that is specified in ARIB STD-B24 and MH-AIT that is specified in ARIB STD-B60 (part of the application loop). Each information element has the meaning in common.

Table 5-2 shows the syntax of the Application element, and Fig.5-2 shows the structure of the Application element.

Table 5-2: Syntax of Application element

```
<xsd:complexType name="Application">
  <xsd:sequence>
    <xsd:element name="applicationIdentifier" type="isdb2:ApplicationIdentifier"/>
    <xsd:element name="applicationDescriptor" type="isdb2:ApplicationDescriptor"/>
    <xsd:element name="applicationTransport" type="isdb2:TransportProtocolDescriptorType"
maxOccurs="unbounded"/>
    <xsd:element name="applicationLocation"
type="mhp:SimpleApplicationLocationDescriptorType"/>
    <xsd:element name="autostartPriorityDescriptor"
type="isdb2:AutostartPriorityDescriptorType" minOccurs="0"/>
    <xsd:element name="cacheControlInfoDescriptor"
type="isdb2:CacheControlInfoDescriptorType" minOccurs="0"/>
    <xsd:element name="randomizedLatencyDescriptor"
type="isdb2:RandomizedLatencyDescriptorType" minOccurs="0"/>
    <xsd:element name="applicationBoundaryAndPermissionDescriptor"
```

```
type="isdb2:ApplicationBoundaryAndPermissionDescriptorType" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

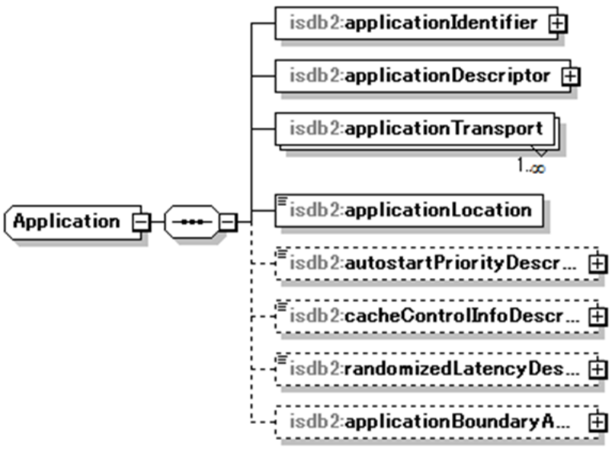
Application
isdb2:applicationIdentifier
isdb2:applicationDescriptor
isdb2:applicationTransport
1..∞
isdb2:applicationLocation
isdb2:autostartPriorityDescr...
isdb2:cacheControlInfoDescr...
isdb2:randomizedLatencyDes...
isdb2:applicationBoundaryA...

Fig.5-2: Structure of Application Element

5.3.2 ApplicationIdentifier element

The following XML schema is additionally applied as an ApplicationIdentifier element. This element responds to the identification of the applications that are specified in ARIB STD-B24 and 0. Each information element has the meaning in common.

Table 5-3 shows the syntax of the ApplicationIdentifier element, and Fig.5-3 shows the structure of the ApplicationIdentifier element.

Table 5-3: Syntax of ApplicationIdentifier element

```
<xsd:complexType name="ApplicationIdentifier">
  <xsd:sequence>
    <xsd:element name="orgId" type="xsd:unsignedShort"/>
    <xsd:element name="appId" type="xsd:unsignedInt"/>
  </xsd:sequence>
</xsd:complexType>
```

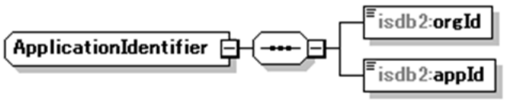

Fig. 5-3: Structure of ApplicationIdentifier

5.3.3 ApplicationDescriptor element

The following XML schema is additionally applied as an ApplicationDescriptor element. This element responds to part of the section structure of AIT (part of the application loop) that is specified in ARIB STD-B24 and the application descriptor, and part of the structure of MH-AIT (part of the application loop) that is specified in ARIB STD-B60 and the MH-application descriptor. Each information element has the meaning in common. ARIB-HTML5 is introduced as the ApplicationType element to be used in this standard.

Table 5-4 shows the syntax of the ApplicationDescriptor element, and Fig.5-4 shows the structure of the ApplicationDescriptor element.

Table 5-4: Syntax of ApplicationDescriptor

```
<xsd:simpleType name="Isdb2ApplicationType">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="ARIB-HTML5"/>
 </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="ApplicationType">
 <xsd:choice>
  <xsd:element name="Isdb2App" type="isdb2:Isdb2ApplicationType"/>
  <xsd:element name="OtherApp" type="mpeg7:mimeType"/>
 </xsd:choice>
</xsd:complexType>
<xsd:complexType name="ApplicationDescriptor">
 <xsd:sequence>
  <xsd:element name="type" type="isdb2:ApplicationType"/>
  <xsd:element name="controlCode" type="mhp:ApplicationControlCode"/>
  <xsd:element name="visibility" type="mhp:VisibilityDescriptor" minOccurs="0"/>
  <xsd:element name="serviceBound" type="xsd:boolean" default="true" minOccurs="0"/>
  <xsd:element name="priority" type="ipi:Hexadecimal8bit"/>
  <xsd:element name="version" type="ipi:Version"/>
  <xsd:element name="mhpVersion" type="mhp:MhpVersion" minOccurs="0"/>
  <xsd:element name="icon" type="mhp:IconDescriptor" minOccurs="0"/>
  <xsd:element name="storageCapabilities" type="mhp:StorageCapabilities" minOccurs="0"/>
 </xsd:sequence>
</xsd:complexType>
```

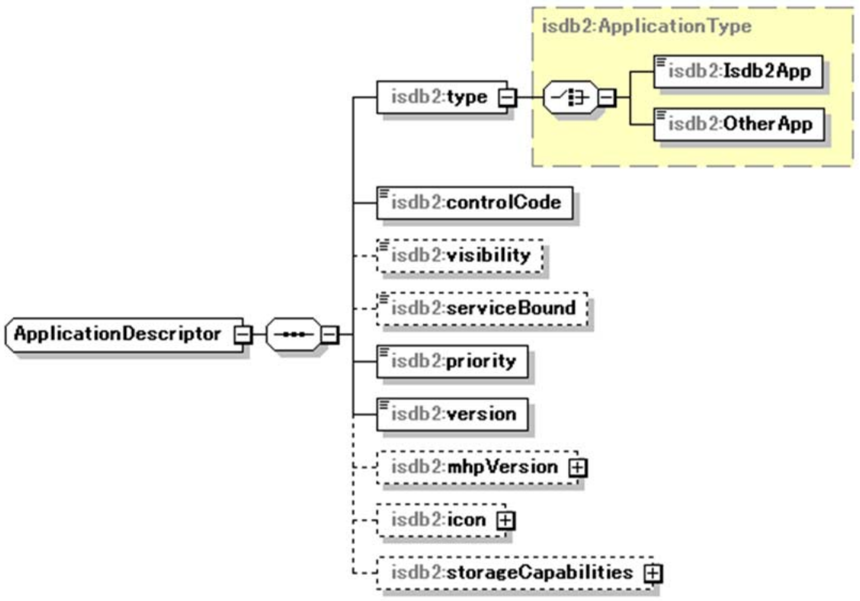

Fig.5-4: Structure of ApplicationDescriptor element

5.3.4 ApplicationTransport element

The following XML schema is additionally applied as an ApplicationTransport element. This element responds to the transport protocol descriptor that is specified in ARIB STD-B24, and part of the MH-transport protocol descriptor that is specified in ARIB STD-B60. Each information element has the meaning in common.

Table 5-5 shows the syntax of the ApplicationTransport element, and Fig.5-5 shows the structure of the ApplicationTransport element.

Table 5-5: Syntax of ApplicationTransport

```
<xsd:complexType name="HTTPTransportType">
  <xsd:complexContent>
   <xsd:extension base="isdb2:TransportProtocolDescriptorType">
    <xsd:sequence>
     <xsd:element name="URLBase" type="xsd:anyURI"/>
     <xsd:element name="URLExtension" type="xsd:anyURI" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
   </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ComponentTagType">
  <xsd:attribute name="ComponentTag" type="ipi:Hexadecimal8bit"/>
</xsd:complexType>
<xsd:complexType name="DCTransportType">
  <xsd:complexContent>
   <xsd:extension base="isdb2:TransportProtocolDescriptorType">
    <xsd:sequence>
     <xsd:element name="DvbTriplet" type="ipi:DVBTriplet"/>
     <xsd:element name="ComponentTag" type="isdb2:ComponentTagType"/>
```

```
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="MMTTransportType">
    <xsd:complexContent>
      <xsd:extension base="isdb2:TransportProtocolDescriptorType">
        <xsd:sequence>
          <xsd:element name="URLBase" type="xsd:anyURI"/>
          <xsd:element name="URLExtension" type="xsd:anyURI" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="TransportProtocolDescriptorType" abstract="true"/>
```

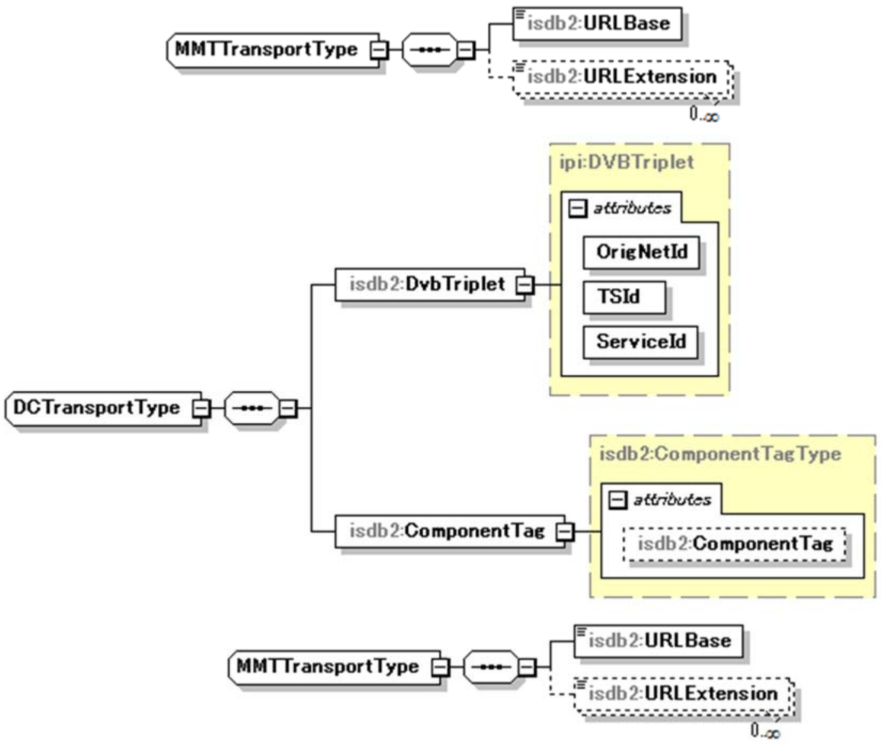

Fig.5-5: Structure of ApplicationTransport element

5.3.5 ApplicationBoundaryAndPermissionDescriptor element

The following XML schema is additionally applied as an ApplicationBoundaryAndPermissionDescriptor element. This element responds to the application boundary and permission descriptor that is specified in ARIB STD-B24, and the MH-application boundary and permission descriptor that is specified in ARIB STD-B60. Each information element has the meaning in common.

Table 5-6 shows the syntax of the ApplicationBoundaryAndPermissionDescriptor element, and Fig.5-6 shows the structure of the ApplicationBoundaryAndPermissionDescriptor element.

Table 5-6: Syntax of ApplicationBoundaryAndPermissionDescriptor element

```
<xsd:complexType name="BoundaryAndPermissionType">
  <xsd:sequence>
    <xsd:element name="permissionBitmap" type="ipi:Hexadecimal16bit" maxOccurs="unbounded"/>
    <xsd:element name="managedURL" type="xsd:anyURI" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ApplicationBoundaryAndPermissionDescriptorType">
  <xsd:sequence>
    <xsd:element name="boundaryAndPermission" type="isdb2:BoundaryAndPermissionType"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

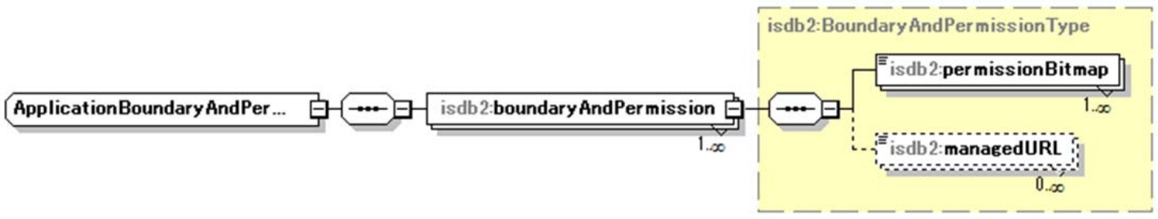
isdb2:BoundaryAndPermissionType
ApplicationBoundaryAndPer...
isdb2:boundaryAndPermission
1..∞
isdb2:permissionBitmap
1..∞
isdb2:managedURL
0..∞

Fig.5-6: Structure of ApplicationBoundaryAndPermissionDescriptor element

5.3.6 AutostartPriorityDescriptor element

The following XML schema is additionally applied as an AutostartPriorityDescriptor element. This element responds to the autostart priority descriptor that is specified in ARIB STD-B24, and the MH-autostart priority descriptor that is specified in ARIB STD-B60. Each information element has the meaning in common.

Table 5-7 shows the syntax of the AutostartPriorityDescriptor element, and Fig.5-7 shows the structure of the AutostartPriorityDescriptor element.

Table 5-7: Syntax of AutostartPriorityDescriptor element

```
<xsd:complexType name="AutostartPriorityDescriptorType">
  <xsd:simpleContent>
   <xsd:extension base="xsd:string">
     <xsd:attribute name="autostartPriority" type="xsd:unsignedShort" use="required"/>
   </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

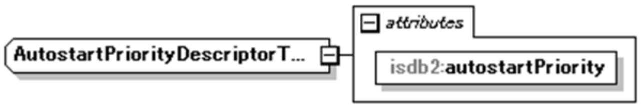

Fig.5-7: Structure of AutostartPriorityDescriptor element

5.3.7 CacheControlInfoDescriptor element

The following XML schema is additionally applied as a CacheControlInfoDescriptor element. This element responds to the cache control info descriptor that is specified in ARIB STD-B24, and the MH-cache control info descriptor that is specified in ARIB STD-B60. Each information element has the meaning in common.

Table 5-8 shows the syntax of the CacheControlInfoDescriptor element, and Fig.5-8 shows the structure of the CacheControlInfoDescriptor element.

Table 5-8: Syntax of CacheControlInfoDescriptor element

```
<xsd:complexType name="CacheControlInfoDescriptorType">
  <xsd:simpleContent>
   <xsd:extension base="xsd:string">
     <xsd:attribute name="applicationSize" type="xsd:unsignedShort" use="required"/>
     <xsd:attribute name="cachePriority" type="xsd:unsignedShort" use="required"/>
     <xsd:attribute name="packageFlag" type="xsd:boolean" use="required"/>
     <xsd:attribute name="applicationVersion" type="xsd:unsignedShort" use="required"/>
     <xsd:attribute name="expireDate" type="xsd:string" use="required"/>
   </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

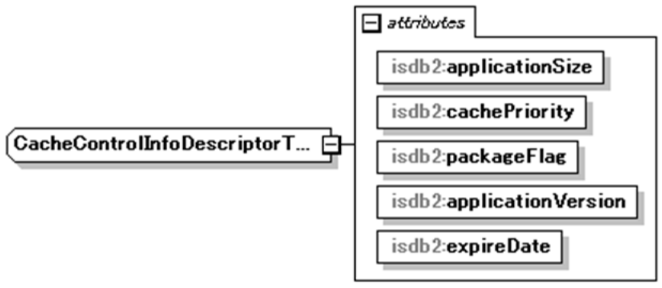

Fig.5-8: Structure of CacheControlInfoDescriptor Element

5.3.8 RandomizedLatencyDescriptor element

The following XML schema is additionally applied as a RandomizedLatencyDescriptor element. This element responds to the randomized latency descriptor that is specified in ARIB STD-B24, and the MH-randomized latency descriptor that is specified in ARIB STD-B60. Each information element has the meaning in common.

Table 5-9 shows the syntax of the RandomizedLatencyDescriptor element, and Fig.5-9 shows the structure of the RandomizedLatencyDescriptor element.

Table 5-9: Syntax of RandomizedLatencyDescriptor element

```
<xsd:complexType name="RandomizedLatencyDescriptorType">
  <xsd:simpleContent>
   <xsd:extension base="xsd:string">
     <xsd:attribute name="range" type="xsd:unsignedInt" use="required"/>
```

```
      <xsd:attribute name="rate" type="xsd:unsignedInt" use="required"/>
      <xsd:attribute name="randomizationEndTime" type="xsd:string" use="optional"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

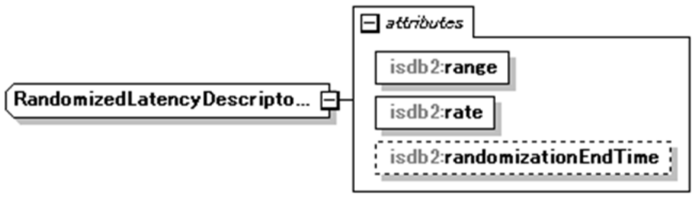

Fig.5-9: Structure of RandomizedLatencyDescriptor element

5.3.9 XML schema of entire AIT of XML format

Table 5-10 shows the XML schema of the entire AIT of XML format.

Table 5-10: XML schema of Entire AIT of XML format

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:ipi="urn:dvb:metadata:iptv:sdns:2008-1"
xmlns:mpeg7="urn:tva:mpeg7:2005"
xmlns:mhp="urn:dvb:mhp:2009"
xmlns:isdb2="urn:arib:isdb2:2014"
targetNamespace="urn:arib:isdb2:2014"
elementFormDefault="qualified" attributeFormDefault="qualified">
<xsd:import namespace="urn:dvb:mhp:2009"
schemaLocation="imports/mis_xmlait.xsd"/>
<xsd:import namespace="urn:dvb:metadata:iptv:sdns:2008-1"
schemaLocation="imports/sdns_v1.4r10_modded.xsd"/>
<xsd:import namespace="urn:tva:mpeg7:2005"
schemaLocation="imports/tva_mpeg7.xsd"/>
<xsd:simpleType name="Isdb2ApplicationType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="ARIB-HTML5"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="ApplicationType">
  <xsd:choice>
    <xsd:element name="Isdb2App" type="isdb2:Isdb2ApplicationType"/>
    <xsd:element name="OtherApp" type="mpeg7:mimeType"/>
  </xsd:choice>
</xsd:complexType>
<xsd:complexType name="ApplicationDescriptor">
  <xsd:sequence>
    <xsd:element name="type" type="isdb2:ApplicationType"/>
    <xsd:element name="controlCode" type="mhp:ApplicationControlCode"/>
    <xsd:element name="visibility" type="mhp:VisibilityDescriptor" minOccurs="0"/>
    <xsd:element name="serviceBound" type="xsd:boolean" default="true" minOccurs="0"/>
    <xsd:element name="priority" type="ipi:Hexadecimal8bit"/>
    <xsd:element name="version" type="ipi:Version"/>
    <xsd:element name="mhpVersion" type="mhp:MhpVersion" minOccurs="0"/>
```

```
      <xsd:element name="icon" type="mhp:IconDescriptor" minOccurs="0"/>
      <xsd:element name="storageCapabilities" type="mhp:StorageCapabilities" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="HTTPTransportType">
    <xsd:complexContent>
      <xsd:extension base="isdb2:TransportProtocolDescriptorType">
        <xsd:sequence>
          <xsd:element name="URLBase" type="xsd:anyURI"/>
          <xsd:element name="URLExtension" type="xsd:anyURI" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ComponentTagType">
    <xsd:attribute name="ComponentTag" type="ipi:Hexadecimal8bit"/>
  </xsd:complexType>
   <xsd:complexType name="DCTransportType">
    <xsd:complexContent>
      <xsd:extension base="isdb2:TransportProtocolDescriptorType">
        <xsd:sequence>
          <xsd:element name="DvbTriplet" type="ipi:DVBTriplet"/>
          <xsd:element name="ComponentTag" type="isdb2:ComponentTagType"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="MMTTransportType">
    <xsd:complexContent>
      <xsd:extension base="isdb2:TransportProtocolDescriptorType">
        <xsd:sequence>
          <xsd:element name="URLBase" type="xsd:anyURI"/>
          <xsd:element name="URLExtension" type="xsd:anyURI" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="TransportProtocolDescriptorType" abstract="true"/>
  <xsd:complexType name="AutostartPriorityDescriptorType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="autostartPriority" type="xsd:unsignedShort" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:complexType name="CacheControlInfoDescriptorType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="applicationSize" type="xsd:unsignedShort" use="required"/>
        <xsd:attribute name="cachePriority" type="xsd:unsignedShort" use="required"/>
        <xsd:attribute name="packageFlag" type="xsd:boolean" use="required"/>
        <xsd:attribute name="applicationVersion" type="xsd:unsignedShort" use="required"/>
        <xsd:attribute name="expireDate" type="xsd:string" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
```

```
</xsd:complexType>
<xsd:complexType name="RandomizedLatencyDescriptorType">
  <xsd:simpleContent>
   <xsd:extension base="xsd:string">
    <xsd:attribute name="range" type="xsd:unsignedInt" use="required"/>
    <xsd:attribute name="rate" type="xsd:unsignedInt" use="required"/>
    <xsd:attribute name="randomizationEndTime" type="xsd:string" use="optional"/>
   </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="BoundaryAndPermissionType">
  <xsd:sequence>
   <xsd:element name="permissionBitmap" type="ipi:Hexadecimal16bit" maxOccurs="unbounded"/>
   <xsd:element name="managedURL" type="xsd:anyURI" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ApplicationBoundaryAndPermissionDescriptorType">
  <xsd:sequence>
   <xsd:element name="boundaryAndPermission" type="isdb2:BoundaryAndPermissionType"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ApplicationIdentifier">
<xsd:sequence>
<xsd:element name="orgId" type="xsd:unsignedShort"/>
<xsd:element name="appId" type="xsd:unsignedInt"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="Application">
  <xsd:sequence>
   <xsd:element name="applicationIdentifier" type="isdb2:ApplicationIdentifier"/>
   <xsd:element name="applicationDescriptor" type="isdb2:ApplicationDescriptor"/>
   <xsd:element name="applicationTransport" type="isdb2:TransportProtocolDescriptorType"
maxOccurs="unbounded"/>
   <xsd:element name="applicationLocation" type="mhp:SimpleApplicationLocationDescriptorType"/>
   <xsd:element name="autostartPriorityDescriptor" type="isdb2:AutostartPriorityDescriptorType"
minOccurs="0"/>
   <xsd:element name="cacheControlInfoDescriptor" type="isdb2:CacheControlInfoDescriptorType"
minOccurs="0"/>
   <xsd:element name="randomizedLatencyDescriptor" type="isdb2:RandomizedLatencyDescriptorType"
minOccurs="0"/>
   <xsd:element name="applicationBoundaryAndPermissionDescriptor"
type="isdb2:ApplicationBoundaryAndPermissionDescriptorType" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ApplicationOfferingType">
  <xsd:complexContent>
   <xsd:extension base="ipi:OfferingBase">
    <xsd:sequence>
     <xsd:element name="ApplicationList" maxOccurs="unbounded">
      <xsd:complexType>
       <xsd:sequence>
        <xsd:element name="Application" type="isdb2:Application" maxOccurs="unbounded"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:element>
```

```
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd: element name="ServiceDiscovery">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element name="ApplicationDiscovery" type="isdb2:ApplicationOfferingType"
maxOccurs="unbounded"/>
    </xsd:choice>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Description

1 Code example of broadcast audio/video object

Each method and code examples of param elements in the broadcast audio/video objects that are specified in 2.2 are shown as follows (MPEG-2 TS case).

Code example of enableFullscreen function

```
var video = document.getElementById('video');

if ('function' === typeof(video.enableFullscreen)) {
  video.enableFullscreen();
} else {
  /* Width, height, z-index are overwritten to bring the element to the foreground
and display it in full screen */
}
```

Code example of disableFullscreen function

```
var video = document.getElementById('video');

if ('function' === typeof(video.disableFullscreen)) {
  video.disableFullscreen();
} else {
  /*Width, height, z-index are overwritten to bring the element back to its
original size */
}
```

Code example of is Fullscreen function

```
var video = document.getElementById('video');

if ('function' === typeof(video.isFullscreen)) {
  if (video.isFullscreen()) {
      /* In case of full screen display */
  } else {
      /* In case of normal display */
  }
}
```

Code example of enableAudioMute function

```
var video = document.getElementById('video');

if ('function' === typeof(video.enableAudioMute)) {
  video.enableAudioMute();
} else {
  /* When muting is not possible */
}
```

Code example of disableAudioMute function

```
var video = document.getElementById('video');

if ('function' === typeof(video.disableAudioMute)) {
  video.disableAudioMute();
} else {
  /* When muting diabling is not possible */
}
```

Code example of isAudioMute function

```
var video = document.getElementById('video');

if ('function' === typeof(video.isAudioMute)) {
  if (video.isAudioMute()) {
      /* When in muting state */
  } else {
      /* When audio output is possible */
  }
}
```

Code example of setAudioSrc function

```
var video = document.getElementById('video');

if ('function' === typeof(video.setAudioSrc)) {
  /* When specifying the second audio channel in a dual monoral audio stream
*/
  video.setAudioSrc("arib://-1.-1.-1/-1;2");
} else {
  /* When src specification of audio is not possible */
}
```

Code example of setVideoSrc function

```
var video = document.getElementById('video');

if ('function' === typeof(video.setVideoSrc)) {
  video.setVideoSrc("arib://-1.-1.-1/-1");
} else {
  /* When src specification of video is not possible */
}
```

Code example of getAudioSrc function

```
var video = document.getElementById('video');

if ('function' === typeof(video.getAudioSrc)) {
  var src = video.getAudioSrc();
  /* processing pursuant to src information*/
} else {
  /* When acquisition of audio src is not possible */
}
```

Code example of getVideoSrc function

```
var video = document.getElementById('video');

if ('function' === typeof(video.getVideoSrc)) {
  var src = video.getVideoSrc();
  /* processing pursuant to src information*/
} else {
  /* When acquisition of audio src is not possible */
}
```

Code example of setCaptionSrc function

```
var video = document.getElementById('video');

if ('function' === typeof(video.setCaptionSrc)) {
   /* when the first language was specified in the captions on the air */
  video.setCaptionSrc("arib://-1.-1.-1/-1;0");
} else {
  /* when sec specification of caption is not possible */
}
```

Code example of getCaptionComponentURL function

```
var video = document.getElementById('video');

if ('function' === typeof(video.getCaptionComponentUrl)) {
  var url = video.getCaptionComponentUrl();
  /* processing pursuant to the caption component url information */
} else {
  /* When acquisition of caption character is not possible */
}
```

Code example of isCaptionExistent function

```
var video = document.getElementById('video');

if ('function' === typeof(video.isCaptionExistent)) {
  if(video.isCaptionExistent("arib://-1.-1.-1/-1;0")){
  /* First language is being broadcast in the captions on the air */
  } else {
  /* First language is not on the air */
}
```

Code example of setCaptionVisibility function

```
var video = document.getElementById('video');

if ('function' === typeof(video.setCaptionVisibility)) {
 /* Instruct to present the caption on the air */
  video.setCaptionVisibility();
} else {
  /* When setCaptionVisibility is not possible */
}
```

Code example of isCaptionVisible function

```
var video = document.getElementById('video');

if ('function' === typeof(video.isCaptionVisible)) {
  if (video.isCaptionVisible()) {
      /* in caption display state */
  } else {
      /* in caption non-display state */
  }
}
```

Code example of addCaptionListener function

```
var video = document.getElementById('video');

if ('function' === typeof(video.addCaptionListener)) {
  /* When the acquired text is processed by a callback function in the first
language captions on the air */
  video.addCaptionListener(function(captiondata) ,"arib://-1.-1.-1/-1;0");
} else {
  /* When adding the acquired listener in caption text is not possible */
}
```

Code example of removeCaptionListener function

```
var video = document.getElementById('video');

if ('function' === typeof(video.removeCaptionListener)) {
  /* removal of all listeners relevant to the captions on the air */
  video.removeCaptionListener();
} else {
  /* when removal of acquired listeners in the caption text is not possible */
}
```

Description example of initial parameter using param element

```
<object id="video" type="video/x-iptvf-broadcast">
  <param name="video_src" value="arib://-1.-1.-1/-1">
  <param name="audio_src" value="arib://-1.-1.-1/-1;2">
  <param name="audio_mute" value="disable">
  <param name="fullscreen" value="enable">
  <param name="caption_src" value="arib://-1.-1.-1/-1;0">
</object>
```

<Blank Page>

Appendix

-

Reprinted parts from IPTV Forum technical specification in this standard

Part of this standard “Specification for Multimedia Coding scheme” is reprinted from IPTV Forum Specification: IPTVFJ STD-0011 Version 2.0. The reprinted parts are shown in Table 1.

・Regarding these reprinted parts, IPTV Forum Specification: IPTVFJ STD-0011 Version 2.0 is the authentic text.
・The Association of Radio Industries and Businesses (ARIB) assumes the responsibility for this reprint.

Table 1: Reprinted Parts

This Standard		IPTVF STD-0011 Applied Parts	
Chapter/ Section	Title	Chapter/ Section	Title
2.2	Broadcast audio/video object	3.2.	Broadcast audio/video object
3.3.7	Application manager object	3.1.2	Application manager object
3.3.8	Application object	3.1.3	Application object
3.3.9	ApplicationInformationTable object	3.1.4	ApplicationInformationTable object
3.3.10	Capabilities object	3.1.5	Capabilities object
3.3.11	ReceiverDevice object	3.1.6	ReceiverDevice object
3.3.11.2	Obtaining information about the product	3.1.19.1	Obtaining information about the product
3.3.12	Stream event target object	3.1.12	Stream event target object
Description		3.2	Broadcast audio/video object

MULTIMEDIA CODING SPECIFICATION FOR DIGITAL BROADCASTING (SECOND GENERATION) ARIB STANDARD

ARIB STD-B62　VERSION 1.0-E1　(Fascicle 2)
(July 31, 2014)

This Document is based on the ARIB standard of “MULTIMEDIA CODING SPECIFICATION FOR DIGITAL BROADCASTING (SECOND GENERATION)” in Japanese edition and translated into English in January, 2015.

Published by

Association of Radio Industries and Businesses

11F, Nittochi Building
1-4-1 Kasumigaseki, Chiyoda-ku, Tokyo 100-0013, Japan

TEL 81-3-5510-8590
FAX 81-3-3592-1103

Printed in Japan
All rights reserved