IPTV STANDARD

Integrated Broadcast-Broadband System Specification

IPTVFJ STD-0010 Version 2.0

Created on March 22, 2013 (Version 1.0)
Revised on June 29, 2014 (Version 2.0)

IPTV Forum Japan

<Blank>

Note: Although this Specification does not specifically make a reference to industrial property rights that are essential to this Specification, the owners of these essential industrial property rights listed below declare the following: “Although the industrial property rights associated with this Specification, as listed in Table 1 below, are owned by the party indicated in the table, these right owners grant users of this Specification a non-exclusive and indiscriminate license to use the patents listed in the table under appropriate conditions. However, this does not apply to cases where the user of this Specification owns industrial property rights essential to the entire or partial content of this Specification, and asserts these rights.”

Table 1 (Selection of the Item 1 Provision)

Patent applicant	Name of invention	Patent application number, etc.	Notes
Sony Corporation	Receiver Device, Receiving Method, Program and Broadcasting System*1	Patent application 2012-531796	Japan
	Receiver Device, Terminal Device, Control Method, Program, and Communication System*1	WO2012/173060	WO
	Information Processing Device, Information Processing Method and Program*1	PCT/JP2012/007160	WO
	Information Processing Device, Information Processing Method Program, Application Information Table Providing Device, and Application Information Table Providing Method*1	PCT/JP2012/007527	WO
	Receiver Device, Receiving Method, Broadcasting Device, Broadcasting Method, Program and Coordinated Application Control System*1	Patent application 2012-207207	Japan
	Receiver Device, Receiving Method, Transmitter Device, Transmitting Method, and Program*1	Patent application 2012-108135	Japan
	Receiver Device, Receiving	Patent application	Japan

	Method, Broadcasting Device, Broadcasting Method, Program and Coordinated Application Control System*1	2012-009549	
	Transmission Apparatus, AIT Transmission Method and Reception Apparatus*1	PCT/JP2012/005215	WO

*1: Effective from IPTVFJ STD-0010 Version 1.0

<Blank>

Table of Contents

Chapter 1. Objectives and Scope of this Specification

The integrated broadcast-broadband system (hereafter simply referred to as “this system”) specified in this Specification is a system in which the existing system of digital broadcasting is integrated with capabilities provided by a broadband network. This system is intended to integrate digital broadcasting in a flexible manner with additional content, server processing, presentation processing involving multiple devices via a broadband network. To realize this, applications, which are written in HTML, etc., are executed on the TV receiver used in this system.

This Specification defines the requirements, overall configuration, control information, communication methods, source coding schemes, and receiver functions in this system.

Chapter 2. References

All or parts of the specifications listed below constitute, through references in this text, provisions of this Specification. If a version is specified for any of the specifications listed, that version shall be referred to. If no version is specified, the latest version shall be referred to.

- The following sections of IPTVFJ STD-0002 “IPTV STANDARD VOD Specifications Version 1.1”
  4.1 “Video Transmission Protocol Based on RTP/RTSP”
  4.2 “Video Transmission Protocol Based on HTTP”
  Chapter 5 “Content Playback Control Metafile”
  6.2.1 “Multiplexing within Service”
  6.2.2 “Details of Operation of MPEG-2 (system)”
  6.2.3 “Time-stamped TS”
  6.5 “Transmission of Stream for Variable-speed Playback”
  Chapter 7 “DRM Specifications”
  Appendix B “Application Specifications of Marlin IPTV-ES System in DRM Specifications”
- ARIB TR-B14 Volume 2 “Functional Specifications for Digital Terrestrial Television Broadcasting Receivers”
- ARIB TR-B15 Volume 2 “Functional Specifications for BS Digital Satellite Receivers”
- ISO/IEC 10918-1 “Information technology -- Digital compression and coding of continuous-tone still images: Requirements and guidelines”
- ISO/IEC 15948:2004 “Information technology -- Computer graphics and image processing -- Portable Network Graphics (PNG): Functional specification”
- ISO/IEC 10646:2012 “Information technology -- Universal Coded Character Set (UCS)”
- CompuServe “GRAPHICS INTERCHANGE FORMAT Version 89a”
- ETSI TS 102 809 V1.1.1 “Digital Video Broadcasting (DVB); Signalling and carriage of interactive applications and services in hybrid broadcast/broadband environments”
- ARIB STD-B24 Version 5.7 Volume 2 “XML-based Multimedia Coding Scheme”, Volume 4 “Application Control Scheme”

Chapter 3. Terms and Definitions

This Specification uses the following terms and definitions.

Term	Definition
Application	Software that is developed and operated to realize a service on this system, and is executed on receivers.
Application boundary	Boundary of area represented by a collection of domains in which a document transition is allowed within an application.
Application control information	Information required for launching an application, such as application location, and control application lifecycle such as application launch or termination.
Broadcaster	An entity that broadcasts scheduled programs to viewers via radio waves.
Broadcast-oriented managed application	Application that is controlled by application control information included in broadcast signals. It can operate only while a broadcast service is being received. It is allowed to access broadcast resources in accordance with the application control information received.
HTML5	Language for writing an application in compliance with the HTML5 Browser Specification (IPTVFJ STD-0011).
Integrated broadcast-broadband functions	Receiver functions that are needed to provide services in this system.
Integrated broadcast-broadband system	A system to provide services which are based on the integration of digital broadcasting services and additional content, server processing, presentation processing involving multiple devices, etc. enabled by capabilities of broadband networks
Managed application	A software program that has been developed and is being operated to implement an individual service provided by this system. Among applications being executed on the receiver, a managed application is one that has been authenticated by a broadcaster (or a business entity entrusted by a broadcaster).

Term	Definition
Non-broadcast-oriented managed application	A managed application that is allowed access to broadcast resources based on a means, such as application authentication, which is executed without using broadcast signals. This application can be launched by a means that does not use broadcast signals.
Revocation	Mechanism for disabling the launch of a specific application or removing a specific application from the system.
Service provider	An entity that provides integrated broadcast-broadband services. It creates and distributes content and applications to provide these services.
Signaling	Notification or advertisement of an application or information related to it.
Unmanaged application	Application other than a managed application. It is not allowed access to broadcast resources.

Chapter 4. Vacant

It has been determined that the content of this chapter is unnecessary and thus the chapter is left blank.

Chapter 5. System Model

The functional model of the system assumed in this Specification (hereafter referred to as “this system”) is shown in Figure 5-1. The relationship between functional blocks in this system is shown in Figure 5-2.

This Specification assumes the system model shown in Figure 5-1 in order to enable third parties other than broadcasters to become service providers and to develop and distribute applications. This arrangement will promote service expansion. However, how the system should be provided is not specified. For example, the broadcaster may also be the service provider. The relationship between the functions and elements in this system is specified below.

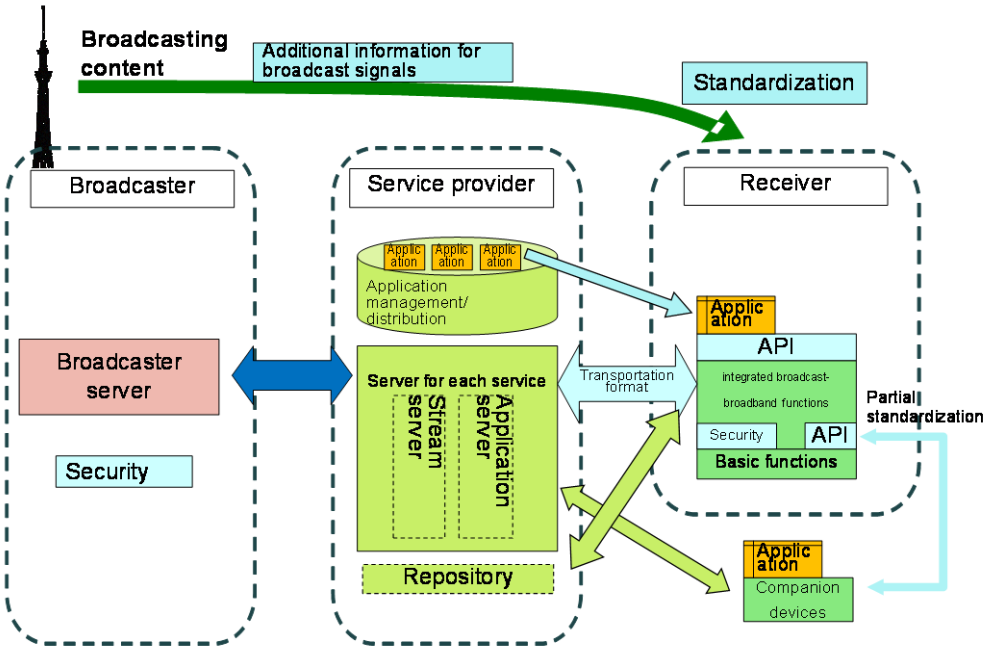


Figure 5-1 Functional model of this system

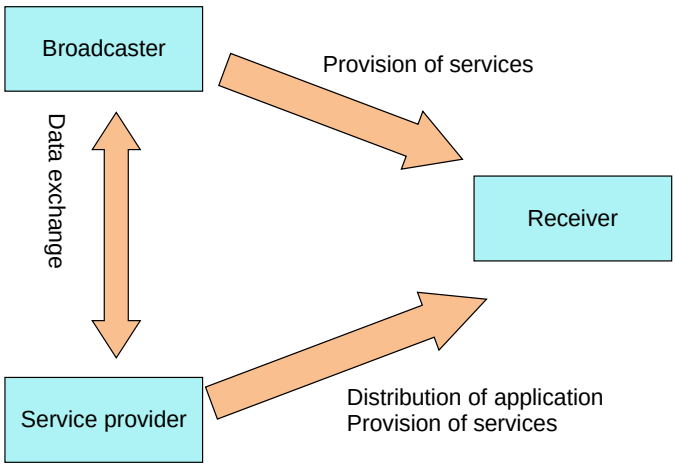


Figure 5-2 Relationship between elements in this system

(1) Broadcaster

This Specification assumes that the broadcaster transmits digital broadcast signals, application control information, and presentation-related control information via broadcast signals. The broadcaster also provides metadata, video content, etc. associated with the broadcast program to service providers in accordance with the relevant contract made between them.

The broadcaster also sends application control information to receivers in order to inform them of applications that work with the broadcast program, and sends instructions for the launch or termination of applications.

Presentation-related control information includes information about how applications are to be superimposed on the broadcast program on a single TV screen and information about whether the presentation of the application concerned is permitted.

The broadcaster operates a broadcaster server, which provides metadata, such as program titles, program IDs, program summaries, names of performers, and broadcasting time and date. It provides metadata to service providers via the API of the broadcaster server.

(2) Service provider

The service provider is an entity that provides services using this system. It produces/distributes content and applications needed for providing the services, and operates servers used for the services. It also conducts sales and makes contracts with users. Different providers may play the role of service providers. For example, a platform operator, such as a broadcaster or a VOD service provider may also act as a service provider. The producer of content or applications does not need to be the distributor of these. The service provider may provide information about links to other service providers.

The API accessed by receivers to implement services is not standardized. It can be freely defined between applications and servers.

To realize the capabilities mentioned above, servers operated by the service provider are broadly classified into two types: servers used to manage/distribute applications and servers specific to individual services.

(A) Servers used to manage/distribute applications
Servers that send applications to receivers.

(B) Servers specific to individual services
Servers that provide the functions required for individual services (VOD program recommendations, multi-language subtitles, social TV, etc.). They not only provide these functions but also distribute content for these services (VOD content, subtitle data, etc.).

(C) Repository
Entity that registers applications for distribution. The repository receives inquiries from receivers, searches the entries to produce a list of applications that can be provided and that match the inquiry, and provides it. It is used by non-broadcast-oriented managed applications.

(3) Receiver

The receiver that supports this system shall be equipped with the functions for implementing integrated broadcast-broadband services (“integrated broadcast-broadband functions” in Figure 5-1) in addition to the functions for receiving existing digital broadcasting. These functions include the following in addition to the function for connecting to a broadband network.

- Function to execute applications in accordance with application control information carried via broadcast signals
- Function to build and control an audio-visual presentation based on the integrated operation of broadcasting and broadband communication
- Function to work with companion devices, etc.

In response to requests from other devices, the receiver’s function to work with companion devices may access broadcast resources, such as program information, or may invoke other receiver functions, such as play control.

Chapter 6. Overview

This chapter specifies the application model, the receiver model and the security model, which are functionally the most important of all the functional blocks specified in the previous chapter.

6.1. Application model

Integrated broadcast-broadband applications are software programs executed on the receiver to realize integrated broadcast-broadband services. In the integrated broadcast-broadband system, applications can be classified from different perspectives, such as whether they work with broadcasting, whether they can be accessed by users anytime, whether their behavior is reliable for broadcasters, and whether they are managed. Considering these various perspectives, this chapter classifies applications and specifies the behavioral model of each category.

6.1.1. Classification of applications

Applications are classified into the following three categories according to how they relate to broadcast services. Refer to Chapter 3 for the definition of each category.

- Broadcast-oriented managed application
- Non-broadcast-oriented managed application
- Unmanaged application

State transitions between these applications are shown in Fig. 6-1.

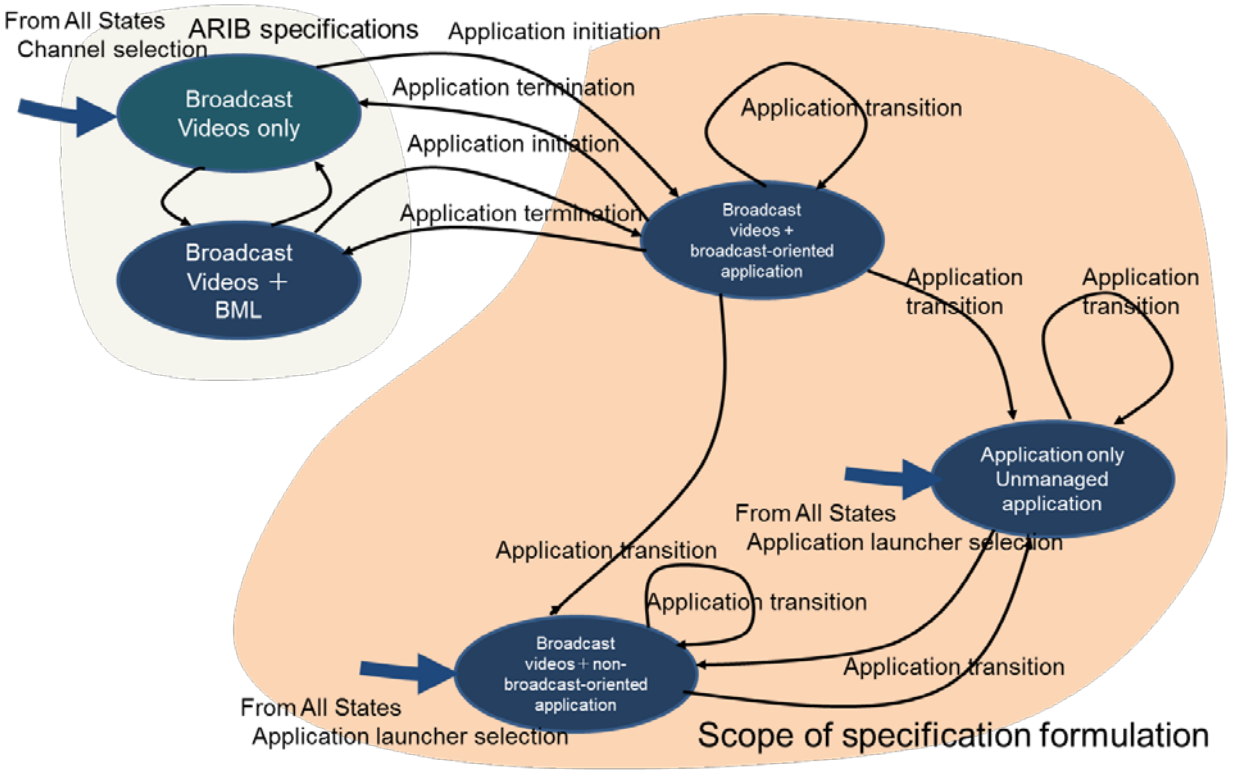


Figure 6-1 State transitions of an application

6.1.2. Application boundary and access to broadcast resources

An application boundary specifies the boundary of the domain in which intra-application document transitions are allowed. An application boundary is defined by the collection of one or more domains (URLs) established for each application based on application control information. Permission to access broadcast resources can be set for each domain (URL) based on application control information. The application boundary and access permissions that have been set can be changed dynamically during application execution. Transitions to outside the application boundary and access to broadcast resources without access permission are prohibited. Receivers shall observe these rules. As long as transitions based on application control information remain within the application boundary, a broadcast-oriented managed application continues to operate until its execution ends or until a transition to another application occurs. The concept of an application boundary does not apply to unmanaged applications because these applications are not controlled based on application control information. Any transition to an unmanaged application is caused using dedicated function calls. When this happens, the application boundary ceases to exist. The concept of an application boundary is shown in Fig. 6-2.

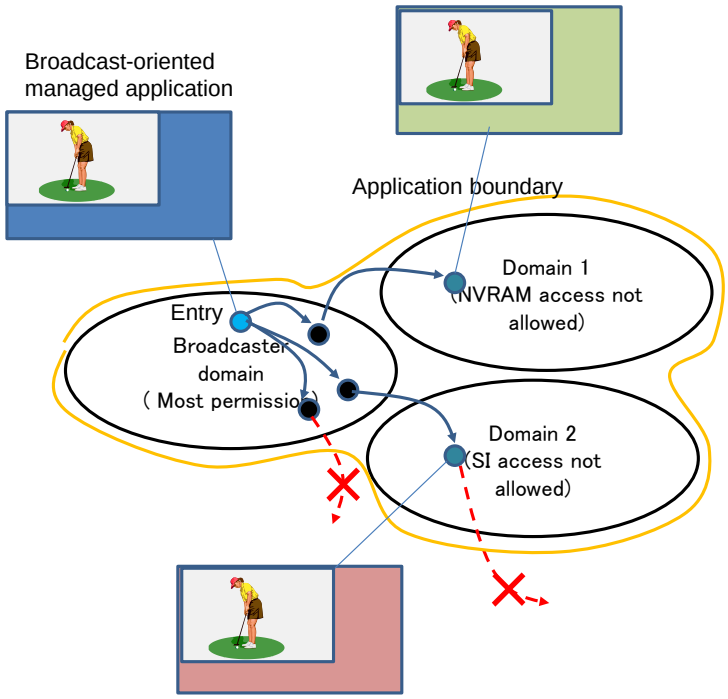


Figure 6-2 Concept of an application boundary

6.2. Receiver model

The functional model of a receiver is described below.

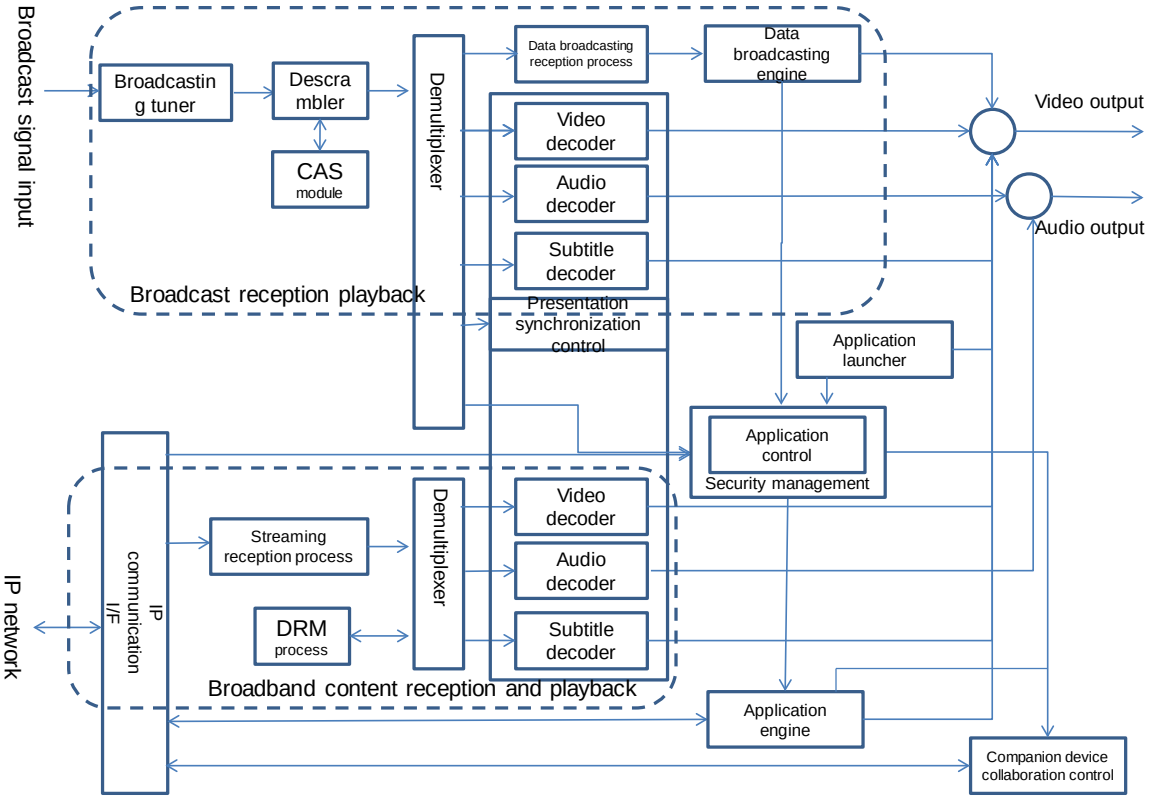


Figure 6-3 Functional model of a receiver

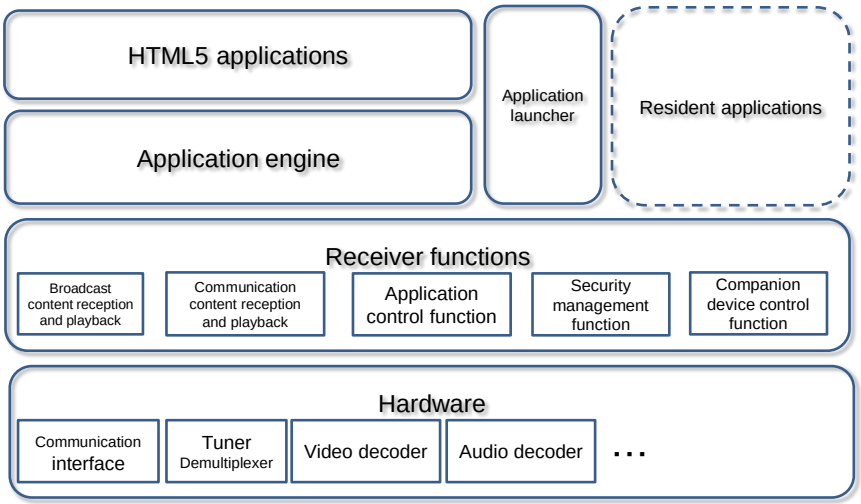


Figure 6-4 Receiver stack model

- A broadcast video reception and play
  A function that receives broadcast signals, selects a specific broadcast service, and plays the videos, audios, subtitles, or data broadcasting content of the selected service synchronously.
- Communication content reception and play
  A function that retrieves video content from a server over a broadband network as a VOD stream, and plays videos, audios, and subtitles of the content.
- Application control
  A function that works with the application engine to control mainly managed applications based on application control information fetched from a server over a broadband network or via broadcast signals. Specifically, it controls and manages the lifecycle and events of each managed application.
- Application engine
  A function that fetches and executes applications. In this Specification, the application engine is an HTML5 browser.
- Device coordinated control
  A function that discovers external companion devices, connects the receiver to the companion devices discovered, and manages coordinated operation between these devices and the applications concerned.
- Security management
  A function that restricts mainly control functions of an application as necessary depending on the application's state and instructions given in control information carried via broadcast signals, etc., in accordance with the specified security management rules.
- Presentation synchronization control
  A function that controls synchronized presentation of the video and audio, etc. of broadcast streams and the video and audio, etc. of broadband streams.
- Application launcher
  A navigation function that enables the user to select and launch mainly non-broadcast-oriented managed applications.

6.3. Security model

6.3.1. Required functions

In order to enhance the convenience for users, it is assumed that applications are provided not only by broadcasters but also by a variety of service providers. To protect the rights on a broadcast video and the interest of viewers, mechanisms for managing applications and controlling the scope of their operations, and mechanisms for rejecting incompatible receivers are required. Security functions

required in the receiver for managing applications for this purpose are shown in Table 6-1.

Table 6-1 Security functions for managing applications

Function	Description
Application authentication	• Handles authenticated applications as managed applications.
Access control	• Allows authorized applications to access broadcast resources (*1) and receiver resources (*2). • Restricts accessible receiver APIs based on the security class of each application.
Presentation control	• Determines the presentation method (*3) of applications in accordance with the intention of broadcasters. • Enables the presentation method of applications to be controlled in the event of an emergency, such as displaying the broadcast program in full screen while making the application temporarily invisible.
Enforcement (*4)	• Makes it possible to prohibit incompatible receivers from accessing applications.
Revocation (*4)	• Disables problematic applications that harm the interests of broadcasters or users.

*1 Broadcast video, audio, metadata SI information, etc.

*2 Video/audio output and network interface, etc., of the receiver.

*3 Whether overlaying an application on the broadcast video is permitted, size of the application display area, and whether the application is displayed.

*4 These functions are expected to be used by non-broadcast-oriented managed applications.

6.3.2. Access control

Among the required functions mentioned above, this section addresses access control. It is

necessary to consider access control because even broadcast-oriented managed applications are not allowed unconditional access to all broadcast resources. Access control is implemented by the application control block shown in Figure . How access control works in broadcast-oriented managed applications is shown in Figure 6-5.

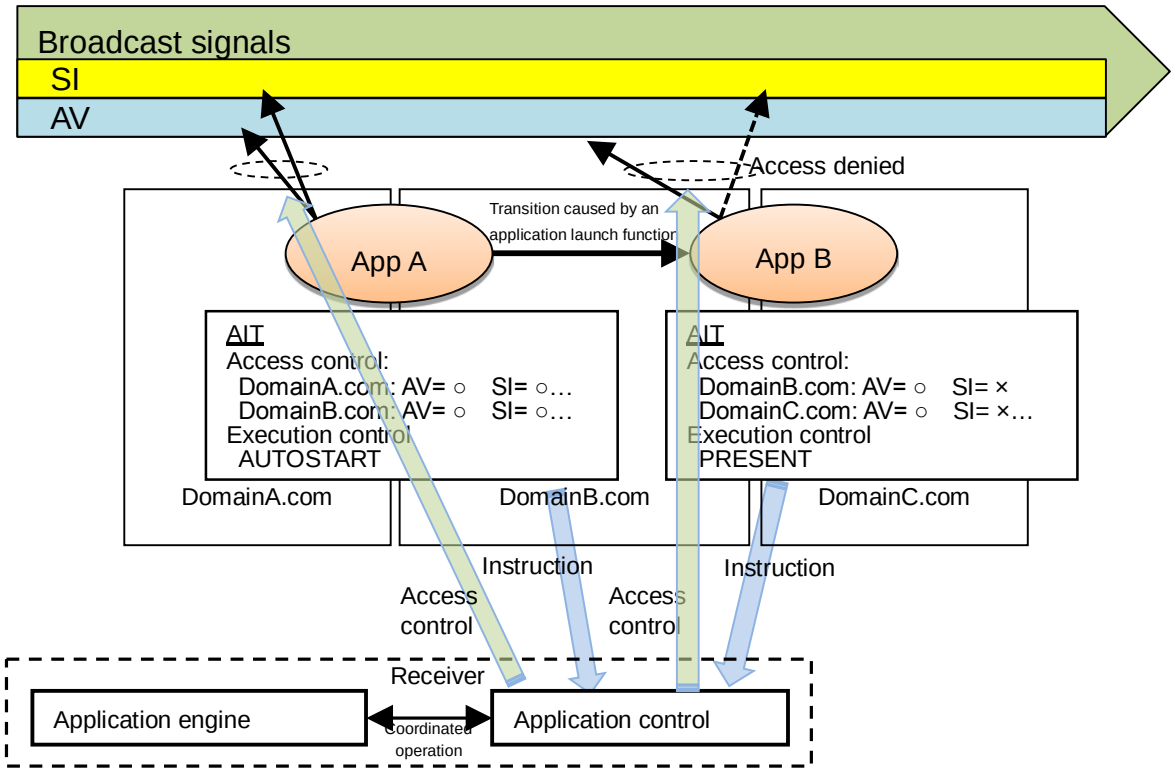


Figure 6-5 Relationship between application boundaries and access control

This figure shows a case where two applications (App A and App B) work with a broadcast program. The execution control of App A is set to AUTOSTART based on application control information (in the application information table (AIT)). So, App A is launched automatically. On the other hand, App B is not launched automatically. App A can make a transition between DomainA.com and DomainB.com while App B can make a transition between DomainB.com and DomainC.com. Both applications are allowed to refer to the video and audio in either of the domains concerned, but only App A is allowed to obtain SI information. When App B is to be launched by an application launch function while App A is accessing DomainB.com, the receiver obtains the AIT for App B and launches App B in accordance with the obtained AIT. Even in such a case, App B is denied access to SI information while App A can access it. The application control block, shown inFigure , in the receiver reads and evaluates the AIT and controls the access through the application engine. In the case of broadcast-oriented managed applications, broadcasters transmit the AIT. As shown in Figure 6-5 and discussed in 6.1.2, access

can be controlled based on the domain (URL) that defines an application boundary.

6.4. Operation model for playing a recorded video

6.4.1. Playing of a recorded video and launch of an associated application

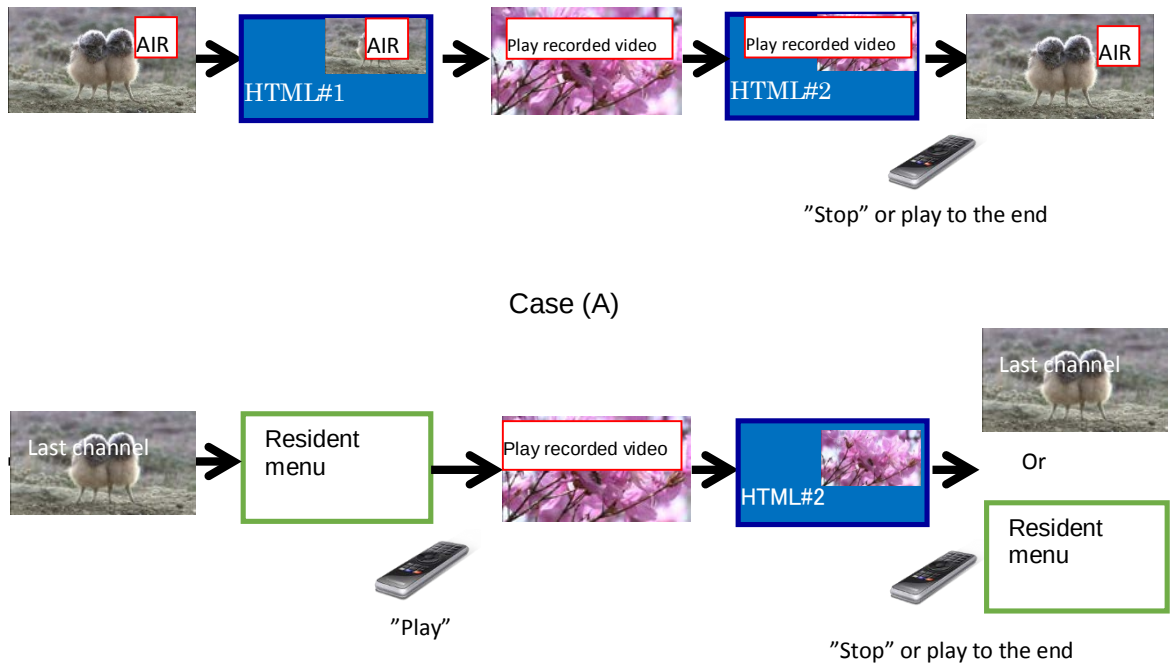


Fig. 6-6 Scenario for launching an application that is associated with playing a recorded video

Fig. 6-6 shows an operation scenario for a case where an application associated with a recorded video is launched when the video is to be played. In Case (A), an application that has been running while a broadcast video is being received specifies and plays a previously recorded video. While the broadcast video is being received, application HTML #1 is launched and plays the recorded video. Application HTML #2, which is associated with the recorded video played, is launched as the recorded video begins to be played. When the playing of the recorded video is stopped, viewing of the broadcast service that was being received earlier is resumed. In Case (B), the receiver specifies and plays a recorded video. Case (B) differs from Case (A) in that the playing of a recorded video is initiated by a receiver function, and that, when the playing is stopped, either the receiver function that has selected the recorded video runs or the reception of the previously received broadcast service is resumed. How the application control information that is used to launch an application is stored depends on the implementation of a particular receiver. The receiver stores the application information itself or the URL from which the information can be fetched, or stores the broadcast signal that contains the application information. Using this information, the receiver refers to the

given application control information and launches the application concerned.

In the operation shown in Fig. 6-6, when the playing of the recorded video is started (i.e., the API specified in IPTVFJ STD-0011 3.1.26.1.4 is invoked), the application that has been executed is stopped in Case (A). In Case (B), the playing of the recorded video starts with full-screen display. In either case, the application execution state just before the start of playing is not stored. This can be equated with selecting a recorded video in a manner similar to selecting a broadcast channel. (For example, the application that has been launched as a result of starting the playing of a recorded video refers to the broadcast video using a broadcast audio/video object specified in IPTVFJ STD-0011 3.2.2.)

6.4.2. Reference by an application to a recorded video to use it as a resource

In cases where an application accesses a recorded video to use it as one of the application's resources, even when the video contains an application that is associated with it, that application is not launched. An example is shown in Fig. 6-7.

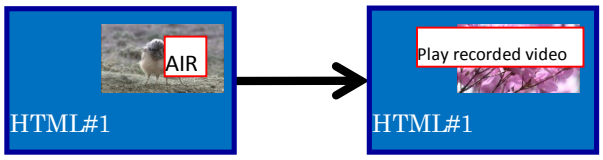


Fig. 6-7 Operation in which an application accesses a recorded video to use the video as one of its resources

In this case, the application that has selected a recorded video as a resource to be displayed and has ordered the playing of the video controls whether the recorded video will be displayed at a reduced size or in full screen. This reference to the resource is made using an API specified in IPTVFJ STD-0011 3.3.2.

6.4.3. Time sequence of recorded video programs

The sequence by which an application refers to a recorded video that is being played is based on the sequence of recording. For example, the time of recording returned by getPVRPlaybackPoint(), which is specified in IPTVFJ STD-0011 3.2.2.1, is the time when this function is executed.

6.4.4. Trick plays

The operation model for trick plays, such as fast-forward, rewind and skip, is specified as follows:

- Applications continue to operate whether the user has selected a trick play or not.
- A trick play changes the playing point. The following is added to handle this change: an API for fetching the playing point, an event to notify the start of a trick play, an event to notify the end of a trick play, and an event to notify that the playing point has come to the end point (EOF).
- Details of how an application processes these key operations shall be specified in the operational rules.

The above operations are shown in Fig. 6-8.

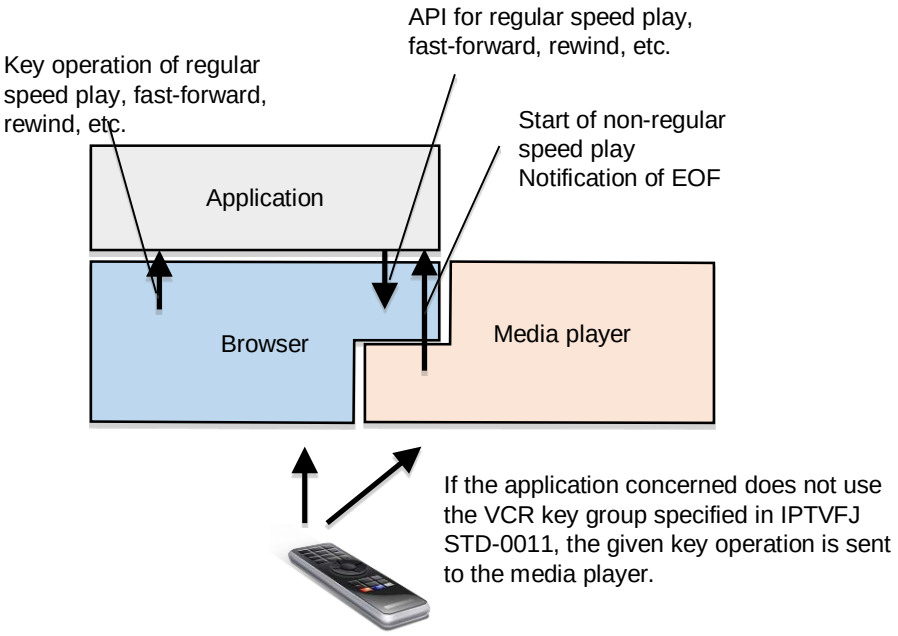


Fig. 6-8 Operation mode for trick plays

If the application concerned uses information about key operations in the VCR key group, the browser sends information about key operations to the application as a key event. The application can execute the operation associated with the key operation using an API specific to each type of key operation. If the application concerned does not use key operations in the VCR key group, information about key operations is sent to the media player that is playing a recorded video, and the media player responds to it directly.

If the user enters a skip command to move the playing point to a specific point, the application sends an instruction to change the playing point. Therefore, it knows the playing point immediately after the skip command. On the other hand, if a fast-forward or rewind operation has been executed

and then playing at regular speed is resumed, the application cannot know the playing point directly. If a trick play has been executed via an application, the application can directly know the time when a trick play is stopped, i.e., when playing at regular speed is started. However, if a key operation for a trick pay is sent directly to the media player, the application cannot tell the time when the trick play is stopped. An event that notifies that a trick play has been started and an event that notifies that a trick play has been stopped are used to enable the application to know this information. When an application detects the ignition of such an event or the start of playing at regular speed, it obtains information about the playing point via an API specific to it, and thus can present information to the user or continue interactive operations with the user.

6.5. Operation model for synchronous high-precision broadcast-broadband display

This section specifies the operation for synchronous display of a broadcast video and a piece of content obtained via broadband communication (broadband content). The operation described here is intended to either draw or play the piece of broadband content continuously in high precision in synchronization with the display of the broadcast video.

The operations for synchronous presentation are classified into two categories according to the type of broadband content to be displayed in synchronization:

1. Synchronization between a broadcast video and an application
   In this category, an application displays data in synchronization with a broadcast video. It is assumed that an application draws characters or graphics continuously in synchronization with changes in the broadcast video. The operation model for this is described in 6.5.1.
2. Synchronization between a broadcast video and a broadband stream
   In this category, a broadband stream that includes either audio or video stream or both is displayed in synchronization with a broadcast video. It is assumed that an application plays, in synchronization with a broadcast video, a video stream that is taken from a different viewing angle. The operation model for this is described in 6.5.2.

The fundamental assumption for broadcast-broadband synchronization specified in this section is that synchronization is based on the reference clock used by digital broadcast systems as the reference time for presenting broadband content.

6.5.1. Operation model for synchronization between a broadcast video and an application

The operation model for an application to display data in synchronization with a broadcast video is described below and shown in Fig. 6-9.

1. The application obtains data, such as characters and graphics that are to be presented in synchronization with a broadcast video. The presentation time is added to the given data based on the reference clock (*1) of the broadcast system. The format and delivery time of the given data depend on the implementation of the given application.
2. The application executes the procedure(*2) for obtaining the current reference clock value.
3. From among the data obtained in 1 above, the application displays data that coincides with the reference clock value obtained in 2. The application may add an application-dependent offset value when associating the data with the reference clock value obtained.
4. Repeat 1 through 3 above. The frequency of this repetition depends on each application.

*1: In a broadcast system using MPEG-2 TS, STC (System Time Clock) is used as the reference clock.
*2: In a broadcast system using MPEG-2 TS, the reference clock value can be obtained using the getSTC function.

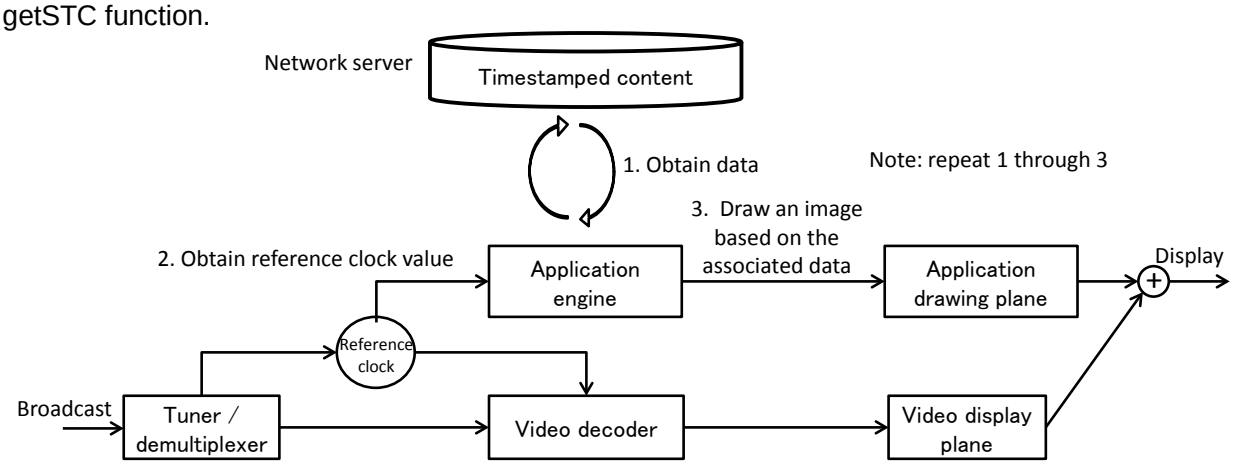


Fig. 6-9 Operation model for synchronization between a broadcast video and an application

6.5.2. Operation model for synchronization between a broadcast video and a broadband stream

The operation model for displaying a broadband stream in synchronization with a broadcast video is described below. The operation model of the receiver for this synchronization is shown in Fig. 6-10. The interactions between the procedure invoked by the application and the playing operation are shown in Fig. 6-11.

1. The application prepares for the playing of a broadband stream while it is displaying a broadcast video. Specifically, it produces a video element, an audio element or an object element, and instructs the receiver to prepare for the playing of the broadband stream.
2. The application executes the addSlave function in the broadcast audio/video object, and instructs the receiver to synchronize the playing of the given broadband stream with the clock of the broadcast stream.
3. The application instructs the receiver to start playing the broadband stream using the play method, etc. of the video element.
4. The delivery server sends the stream, adding to the video data and audio data in the stream the display time information based on the broadcast reference clock. The format of the broadband stream to be delivered shall be determined as described elsewhere in this document and in the operational rules.
5. In response to the instructions 1 to 3 above from the application, the receiver starts receiving a broadband stream and starts playing it in synchronization with the broadcast reference clock.

Specifically, the receiver plays the video and audio data in the broadband stream in such a way that the presentation time attached to each item of data is synchronized with the reference clock. Normally starting of this operation involves buffering of the broadband stream. During this buffering time, the receiver shall continue to present the broadcast video without interruption. As long as the given broadband stream continues, the receiver continues the synchronization of the presentation.

6. When the playing of the broadband stream is to be stopped, the application instructs the receiver to stop the playing of the stream using the pause method, etc., of the video element. When so instructed, the receiver stops playing the broadband stream and continues to present the broadcast video, without interruption.

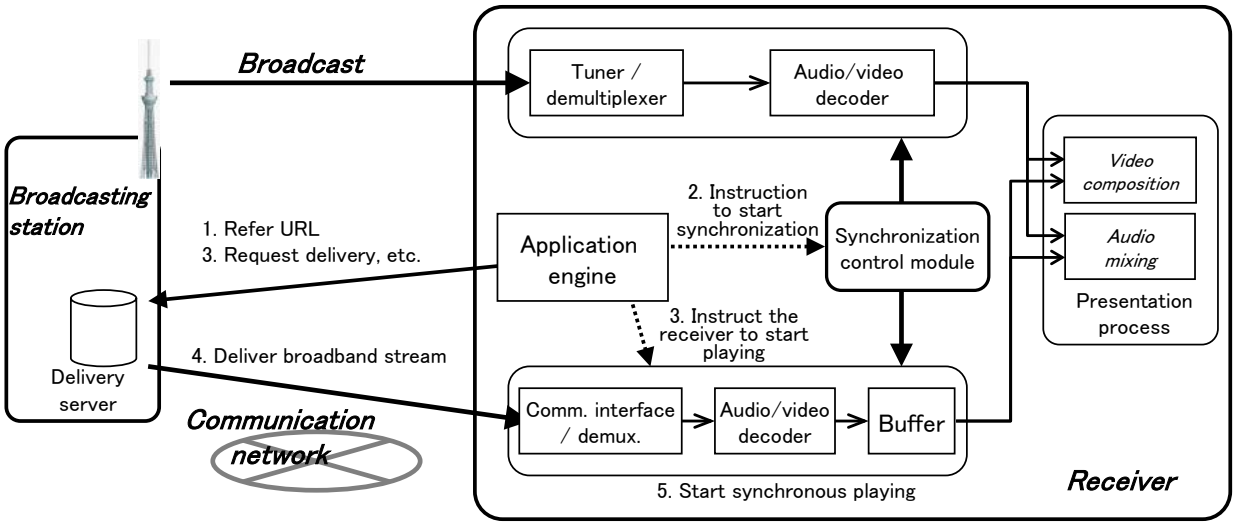


Fig. 6-10 Operation model for synchronization between a broadcast video and broadband stream

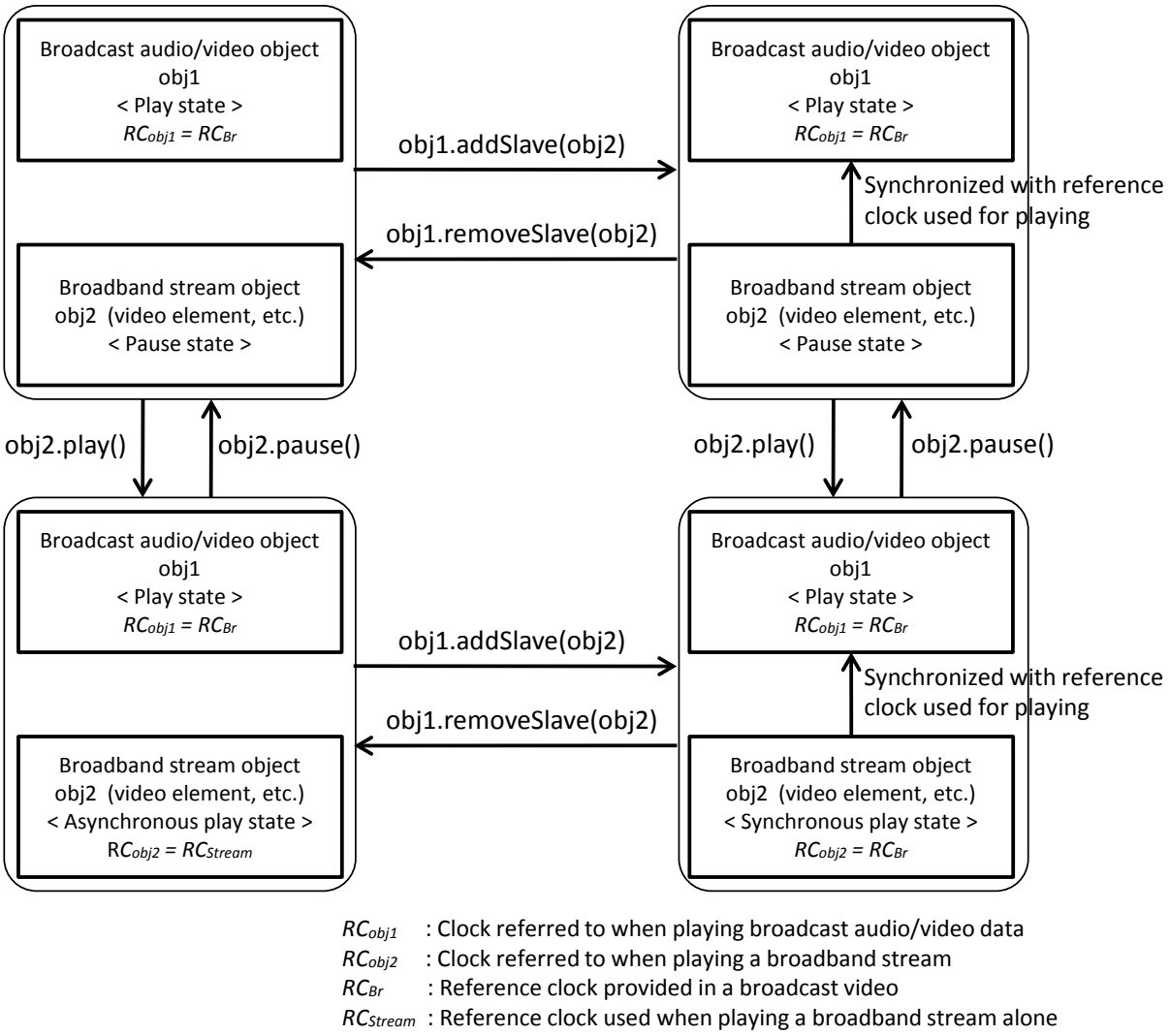


Fig. 6-11 Interactions between the application API for broadcast video/broadband stream synchronization and playing operation

6.5.2.1. Operation model when the buffer for a broadband stream underruns

During synchronous playing (in the state of Step 5 in 6.5.2), it may not be possible to present broadband content normally because it has not been obtained in time before the presentation time on the receiver's reference clock. This can cause a buffer underrun, Even in such a case, presentation of the broadcast video shall not be affected and it shall be continued.

How broadband content is presented in such a case shall be specified in the operational rules.

6.5.2.2. Applying an offset

When an application instructs the receiver to synchronize a broadband stream with the clock of the broadcast video (Step 2 in 6.5.2), it may specify an offset value. In such a case, the receiver

controls the presentation time of the broadband stream based on the reference clock value plus the given offset value.

6.5.3. Synchronous presentation of an MPEG-2 TS-based broadcast video and broadband content

An example of data transmission/reception for implementing broadcast-broadband synchronization as described in 6.5.1 and 6.5.2 in a broadcast system that uses MPEG-2 TS for multiplexing is shown in Fig. 6-12.

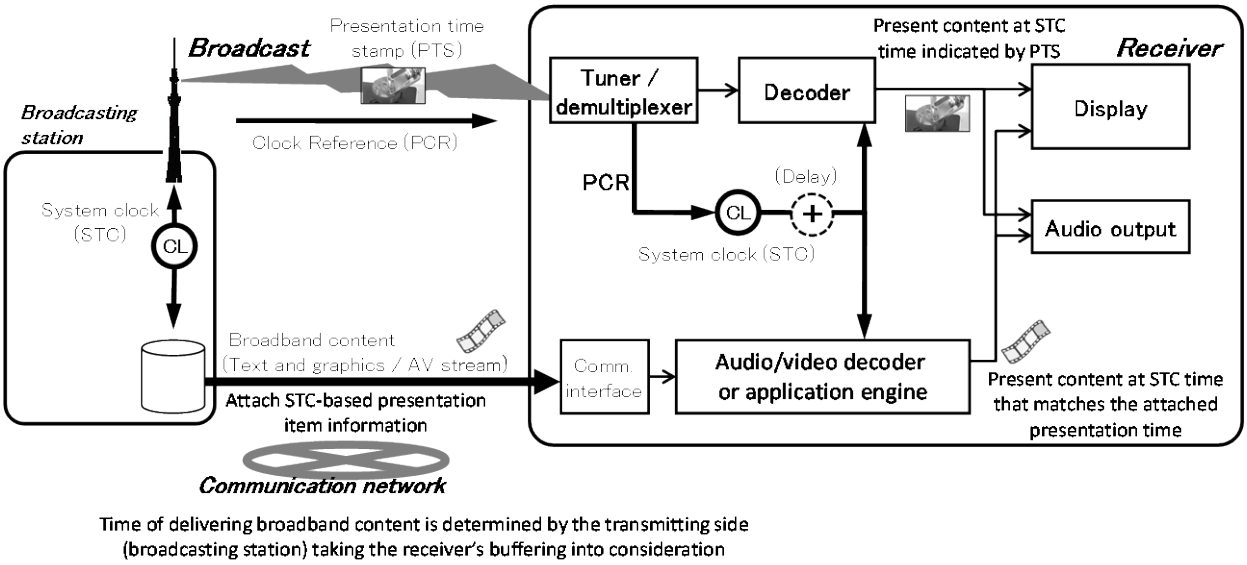


Fig. 6-12 Basic model for synchronous presentation of broadcast and broadband content

The broadcasting station broadcasts a stream as usual based on the system time clock (STC), which serves as a reference clock, multiplexed on the broadcast signal as PCR. For the audio/video data in the broadcast stream, the broadcasting station specifies the presentation time information (PTS) that refers to STC.

When broadband content that synchronizes with a broadcast video is to be delivered, in addition to the above operations, the broadband content delivery server delivers the content attached with the presentation time based on the same system clock as that for the broadcast video. The service provider determines the format and the timing of delivering the broadband content depending on the application. In particular, in the case of synchronizing a broadcast video and a broadband stream, which was described in 6.5.2, the timing of delivering a broadband stream is set by the transmitting side so that buffering is performed appropriately taking the size of the receiver's buffer into consideration. It is assumed that the size of the receiver's buffer will be specified in the operational rules. Note that the network server may be installed in a place other than the broadcasting station.

As the receiver performs normal broadcast reception and presentation, it also receives and presents the broadband content. The receiver presents the broadband content in synchronization with the broadcast video by referring to the system time clock based on PCR received from the broadcast video.

6.5.3.1. Operation model for delaying the presentation of a broadcast video

Depending on the service implementation conditions, it may be necessary to deliver the broadband content with some delay from the broadcast video. It may be difficult to receive and present the given broadband content in time for the required presentation time that is based on the broadcast system time clock. For such a case, we assume that synchronization of the broadband content with the broadcast video may be ensured by shifting the system clock value in the receiver (dotted circle in Fig. 6-12), buffering the broadcast video for a certain appropriate time and then presenting it.

Details for this operation model are to be specified in the future.

Chapter 7. Application Control Information for Broadcast-Oriented Managed Applications

This chapter specifies application control information for broadcast-oriented managed applications. It also explains the concept of controlling applications in accordance with this information, and defines specific control operations.

7.1. Application control information and methods of transmitting this information

7.1.1. Overview of application control information

The aim of application control information is to notify the receiver of the presence of a broadcast-oriented managed application that works with a broadcast service and to instruct the receiver to control the application. The receiver learns of the presence of an application by discovering the presence of a description on the application in application control information. Major items of information included in application control information are shown in Table 7-1.

Table 7-1 Items of information included in application control information

Name	Description
Application type	Describes the application type. Only HTML5 documents are used in this Specification.
Application identifier	This consists of an organization identifier, which identifies the provider, and an application identifier, which identifies an application within the organization.
Application control code	Controls operations on the application. One of the following is specified: ● Automatic start ● Operable ● Termination ● Pre-fetch
Application profile	Value that indicates a receiver function required by the application. This is indicated in combination with optional functions of the receiver. If the receiver has the required function, it determines that it can execute the application.
Application source information	Information that specifies the location of the application (URL, etc.). This information supports both the case where the application exists in a communication network and the case where the application is transmitted via broadcast signals.
Application boundary and access	Specifies one or more domains (URLs) within which the

permission setting	broadcast-oriented managed application is allowed to operate. In addition, sets permission to access broadcast resources for each domain (URL) function by function.
Start priority	Indicates the priority for automatic start among data broadcasting content and other broadcast-oriented managed applications.
Cache control information	Information used to control caching of the application resource in preparation for re-use of that application.
Server access distribution parameter	A parameter set to distribute access attempts to servers by receivers in order to reduce the load on individual servers when receivers attempt to fetch applications.

7.1.2. Detailed specification of application control information

The following two types of format are specified for application control information:

- ✧ Section-format AIT

  Application information table (AIT) written in binary. This format is specified in ARIB STD-B24, but some extensions are added. See A.4.1 for detailed information.

- ✧ XML-format AIT

  Information equivalent to what is in the AIT but written in XML. This is specified in ARIB STD-B24 Volume 4 Chapter 6.

In this Specification, application control information in either of the above formats shall be referred to as AIT.

7.1.3. Method of transmitting application control information

The following three methods are assumed for transmitting application control information.

- ✧ Section transmission of AIT

  A method of transmitting AIT via broadcast signals. AIT in section format is transmitted as a component in the target TV service. In this case, a data coding method descriptor, including ait_identifier_info() specified in ARIB STD-B24 Volume 4 Chapter 7, shall be placed in the ES loop of the PMT that carries the AIT.

- ✧ Data carousel transmission of XML-format or section-format AIT

  A method of transmitting AIT via broadcast signals. AIT in XML format or in section format is transmitted as a component in the target TV service using data carousel transmission specified in ARIB STD-B24 Volume 3. In cases where the ES conveys only application control information, the relevant data coding method descriptor, including ait_identifier_info() specified in ARIB STD-B24 Volume 4 Chapter 7, shall be placed in the relevant ES loop of the PMT. In

cases where application control information is included in a component that also transmits data broadcasting content, no related information is placed in the PMT. Therefore, it is necessary in operation to specify additional rules, such as transmitting AIT in a module of a specific module_id.

- ✧ Delivery of an XML-format or section-format AIT file via a broadband network
  An AIT file in XML format or section format is delivered using the HTTP or HTTPS protocol over the Internet.

7.2. Application unit and identification

7.2.1. Definition of application unit

An application unit is defined as the scope of the broadcast-oriented managed application designated by the application_identifier in AIT. In this Specification, an application unit is a set of HTML documents and the resources referred to by these documents. The entry document of these HTML documents is the HTML document that exists at the location specified by the application source information in application control information. (See Section 7.4 for the method of referring to an application based on application source information.) How the scope of an application unit for a broadcast-oriented managed application is determined may depend on actual operation. In other words, the scope of an application unit can be determined in a variety of way. For example, it may be all applications associated with a specific broadcast program, a specific set of functions, a unit for reuse, or a unit of division of production work. In all cases, it is assumed that one of these units applies lifecycle control based on state transitions, including launch and termination of an application, and caching for reuse.

7.2.2. Application identifier

An application identifier (application_identifier in AIT) identifies an application unit. It consists of organization identification (organization_ide) and application identification (application_id), which are assigned by each provider. This makes it possible to identify an application unit in a globally unique manner. A provider may reassign the same application_id to temporally separate applications. In such a case, assignment of the same ID value to different applications within a receiver shall be avoided by making sure that the caching periods of all the applications except the target one have expired.

7.2.3. Scope of an application unit

Since HTML documents, which constitute a part of an application in this Specification, can inherently be linked to each other endlessly, it is necessary to specify the scope of each application unit unambiguously in application control information. This is specified as follows:

- ➢ The scope specified by the application boundary and permission descriptor in application control information (application boundary and permission descriptor in AIT) is defined as the

application boundary. Any document transitions within this scope are considered to be within the application unit. (Document transitions to outside the application boundary are prohibited.)

- When an explicit transition is initiated from an HTML document in one broadcast-oriented managed application to another broadcast-oriented managed application by calling the replaceApplication() function, this is a transition to outside the application unit even when the entry HTML document of the destination application is located within the boundary of the original application. In this case, the scope of a new application unit is defined for the destination application.
- When an explicit transition is caused from an HTML document in a broadcast-oriented managed application to an unmanaged application by calling the exitFromManagedState() function, the application boundary disappears and the destination application is outside the scope of application control even if the HTML documents of the destination application are located within the boundary of the original application.
- Once an application is launched, and operation of a specific application unit is started from its entry HTML document, all operations are regarded as being within the same application unit until the application is terminated or makes a transition to a different application.

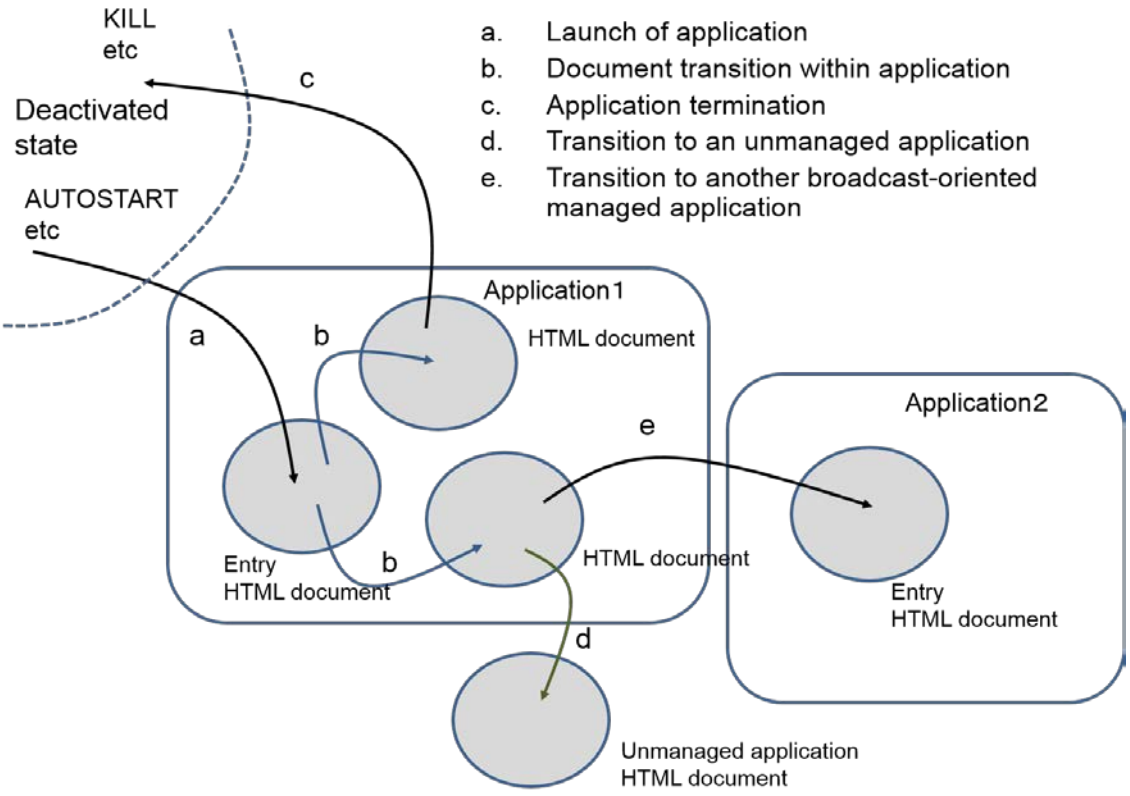


Figure 7-1 Concept of application units

7.3. Application start priority control

If both data broadcasting with BML documents and broadcast-oriented managed applications based on this Specification are simultaneously associated with a broadcast service, it is necessary to be able to specify which should be launched with the higher priority. Furthermore, it may become possible to use multiple broadcast-oriented managed applications (which may be of the same application type or of different application types) in the future. Therefore, it is necessary to consider how to instruct receivers, whose level of compliance with this Specification may vary, clearly with regard to the priorities with which multiple applications should be launched.

7.3.1. Start priority control information

The following items of control information are defined to specify start priority.

- Start priority information for data broadcasting
  Whether data broadcasting should be launched with the highest priority is specified in additional_arib_bxml_info() in the data coding method descriptor that is associated with the component that transmits data broadcasting content on PMT.
- Start priority information for a specific application type
  The order of priority for specific application types is specified in ait_identifier_info() in the data coding method descriptor associated with the component that transmits AIT on PMT.
- Start priority information for a specific application
  The start priority of a specific application (the start priority information descriptor in AIT) is specified in application control information.

7.3.2. Start priority control operations

Start priority control for data broadcasting and broadcast-oriented managed applications in a receiver that is compliant with this Specification follows the sequence described below. Only one broadcast-oriented managed application can be set to AUTOSTART at a time. The following sequence shall take place not only when a channel is selected but also when PMT is re-fetched due to the termination of an application, etc.

(1) The receiver reads data broadcasting start priority information by referring to additional_arib_bxml_info() in the data coding method descriptor associated with the component that transmits data broadcasting content on PMT, and checks whether the data broadcasting has the highest start priority.

(2) If data broadcasting has the highest priority in (1), the receiver shall receive data broadcasting content. Therefore, if AUTOSTART is specified for data broadcasting, the receiver fetches the launch document for data broadcasting from the data carousel, and launches data broadcasting. (If the receiver is already receiving data broadcasting, it

continues to receive it.)

(3) If data broadcasting does not have the highest priority in (1), the next step can be one of the following alternatives:
- If AIT is transmitted in a dedicated component, the receiver checks the start priority information written in ait_identifier_info of the data coding method descriptor associated with the AIT. If the broadcast-oriented managed application has the highest priority, go to (4). Otherwise, go to (5).
- If AIT is not transmitted in a dedicated component, or if AIT is transmitted in a dedicated component but priority is not specified in the start priority information of the data coding method descriptor, the receiver fetches AIT, and refers to the start priority (start priority information descriptor in AIT) associated with the application for which the application control code is set to AUTOSTART. If the broadcast-oriented managed application has the highest priority, go to (4). Otherwise, go to (5).

(4) (For the case where a broadcast-oriented managed application is given the highest priority in (3)) If the receiver has not fetched AIT, it fetches it. The receiver fetches the relevant application from the server specified as the application source, using the reference method described in 7.3.2, or from the broadcast stream. The receiver launches the application.

(5) (For the case where a broadcast-oriented managed application is not given the highest priority in (3)) The receiver receives the data broadcasting. Therefore, if AUTOSTART is specified, the receiver fetches the launch document for the data broadcasting from the data carousel, and launches the data broadcasting. (If the receiver is receiving the data broadcasting, it continues to receive it.)

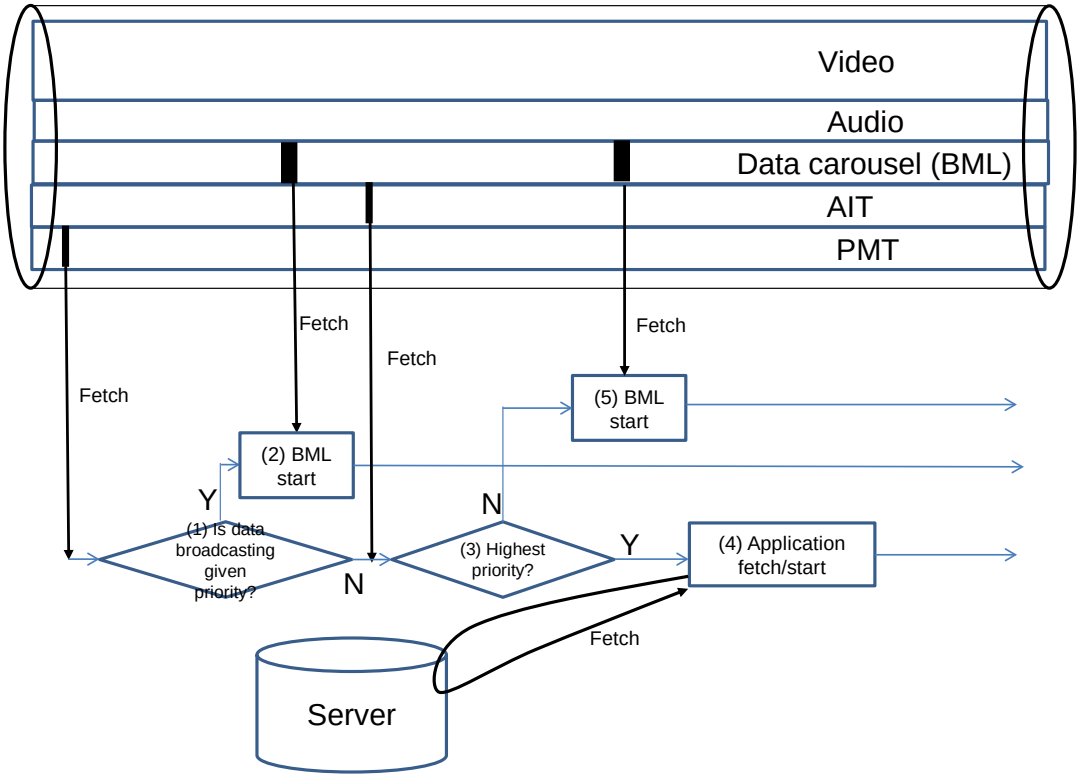


Figure 7-2 Priority control for data broadcasting and a broadcast-oriented managed application

7.4. Fetching an application

The application source specified in application control information is the location information used to fetch the first HTML document to be referred to when the application concerned is launched. Applications may be placed on servers in a communication network, or may be transmitted via broadcast signals. Therefore, the specification of the location information in AIT is designed to support both application fetch via communication and that via broadcast signals.

7.4.1. In the case of fetch via communication

The receiver can fetch the entry file of the target application from a URL created by concatenating the URL base specified in the transport protocol descriptor with its protocol_id indicating HTTP/HTTPS transport in AIT, the URL extension, and the application URL specified in the simple application location descriptor.

Example:

Transport protocol descriptor

URL base: http://www.xbc.co.jp

URL extension: hybrid/appsA

Simplified application location descriptor

Application URL: app1/index.html

Then, the application source URL is:

http://www.xbc.co.jp/ hybrid/appsA/app1/index.html

7.4.2. In the case of fetch via broadcast signals

First, the receiver identifies a component from original_network_id, transport_stream_id, service_id (which is not required if the application is within the same service) and component_tag specified in the transport protocol descriptor with its protocol_id indicating data carousel transmission in AIT. Then, the receiver fetches the specific resource of the specific module from the application URL specified in the simplified application location descriptor. Here, the above application URL must be written in the format <module_name>(/<resource_name>). (If no name descriptor is placed in DII, module_name is written in hexadecimal with “0x” removed from its module_id.)

Example

Transport protocol descriptor

Component tag: 0x40

Simplified application location descriptor

Application URL: 0000/index.html

Application source

A resource called index.html that is included in a module with its module_id= 0x0000 in the data carousel of the component that shows a component tag value of 0x40 in the same service.

7.4.3. In the case where both fetch via communication and fetch via broadcast signals are possible

Both the transport protocol descriptor for fetch via communication and that for fetch via broadcast signals are put in place. The application URL in the simplified application location descriptor shall be specified in the same manner for both cases. Therefore, it is necessary to place on the server a folder whose name corresponds to the module number in the data carousel in a manner suitable for the application specification method used in fetch via broadcast signals. Figure 7-3 shows an example of AIT and application source resolution when both fetch via broadcast signals and fetch via communication are included.

Example

Transport protocol descriptor 1 (fetch via broadcast) transport_stream_label=1

original_network_id= 0x7FE9

transport_stream_id= 0x7FE9

service_id= 0x0448

component_tag= 0x40

Transport protocol descriptor 2 (fetch via communication) transport_stream_label=2

URL base: http://www.xbc.co.jp/

URL extension: apps1

Simplified application location descriptor

Application URL: 0010/index.html

Application source

Broadcast: Resource called index.html that is included in a module with its module_id= -0x0010 in the data carousel of the component that shows a component tag value of 0x40 in the service specified by original_network_id/transport_stream_id/service_id

Communication: application source URL

http://www.xbc.co.jp/apps1/0010/index.html

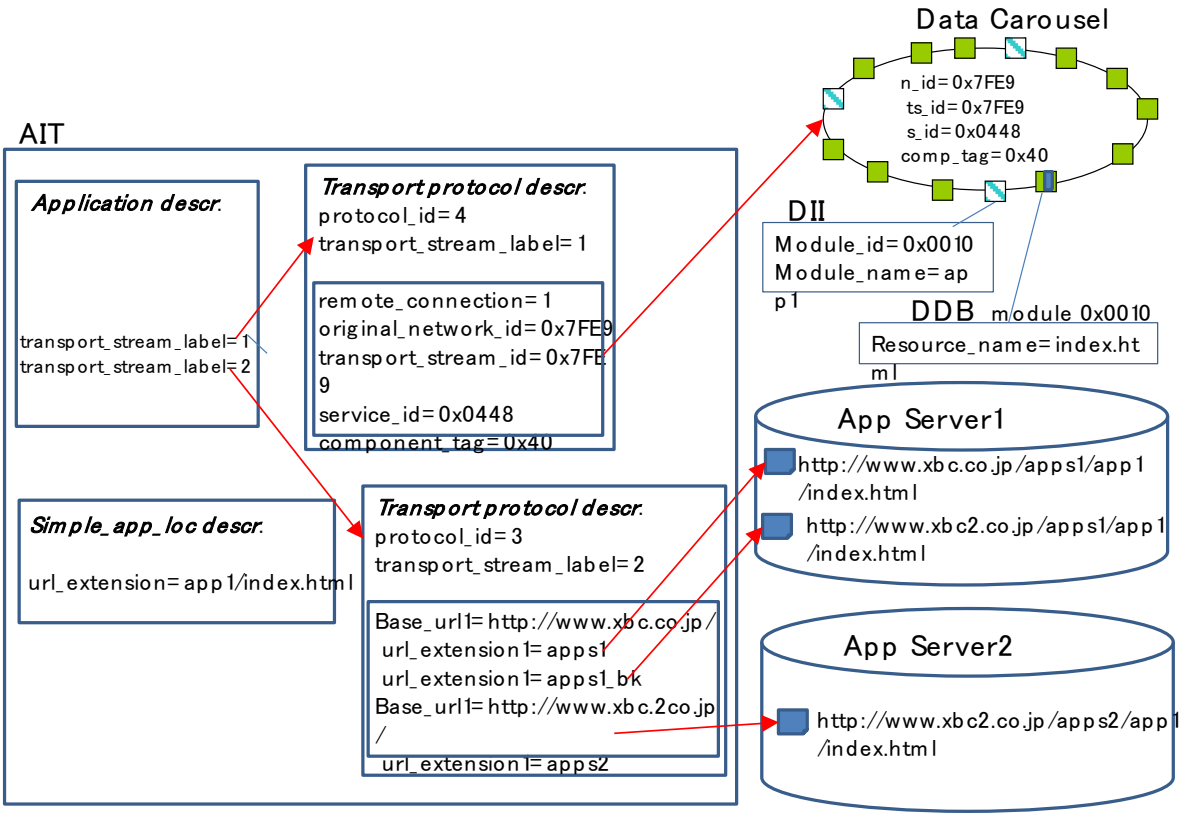


Figure 7-3 Relationship of references in application fetch via broadcasting and of that via communication (example)

7.5. Application lifecycle control

If start priority control specified in 7.3 indicates that a broadcast-oriented managed application has the highest priority, the receiver fetches, launches and terminates the application, and/or makes a transition based on application control information. This section specifies control over such an application lifecycle.

7.5.1. Overview of application lifecycle control

There are two ways of controlling application lifecycle: control based on application control information update (Scheme A) and control based on the application control information referred to at the time of application launch (Scheme B).

7.5.1.1. Scheme A: Control based on application control information update

This scheme is applied when application control information is transmitted via broadcast signals in a transmission method specified in application control information as described in 7.1.3. It should be noted that, even when application control information is transmitted via broadcast signals, this scheme is applied only in cases where the receiver recognizes that application control information is transmitted constantly on PMT, etc. The basic sequence for application lifecycle control in this case is shown in Figure 7-4 (a). First, the receiver needs to recognize the presence of application control information and constantly monitor the information. When the receiver receives application control information (AIT) in which the application control code specifies AUTOSTART while it is receiving a broadcast service, it fetches and launches the specified application. Later, AIT is updated. When the receiver receives AIT in which the application control code indicates termination (KILL), it terminates the ongoing application. A more complicated lifecycle control sequence assuming realistic operation is shown in Figure 7-4 (b). In this case, the receiver receives AIT in which the application control code specifies PREFETCH. This AIT is transmitted so that the receiver will fetch application app1 before the program starts. The receiver fetches app1. AIT is updated. The receiver receives AIT in which the application control code specifies AUTOSTART. This AIT is transmitted from the start of the program. The receiver launches app1, which it has already fetched. Later, when AIT is updated at the time of switching of the ongoing program, the application control code associated with the ongoing application becomes neither AUTOSTART nor PRESENT, and the receiver receives AIT specifying AUTOSTART of another application. The receiver terminates the ongoing app 1, and fetches and launches a new application, app2. Later on, when the user switches to another broadcast service, the receiver terminates the ongoing application, app2. After that, it is assumed that the receiver determines start priority in the new broadcast service, and launches data broadcasting or another broadcast-oriented managed application. When AIT is updated but AUTOSTART of another application is not specified and the application control code associated with the ongoing application becomes neither AUTOSTART nor PRESENT, the receiver terminates the application after a certain period of time due

to time-out.

The receiver can launch an application not only when application control information is updated but also when a function is executed by data broadcasting or by another application. In the latter case, when the receiver executes the function, it fetches and refers to AIT. If the receiver finds that a predefined launch condition is fulfilled, it fetches and launches the application concerned. As explained above, the lifecycle of a broadcast-oriented managed application is realized through control that is related to broadcast signals mainly based on the application control information monitored by the receiver.

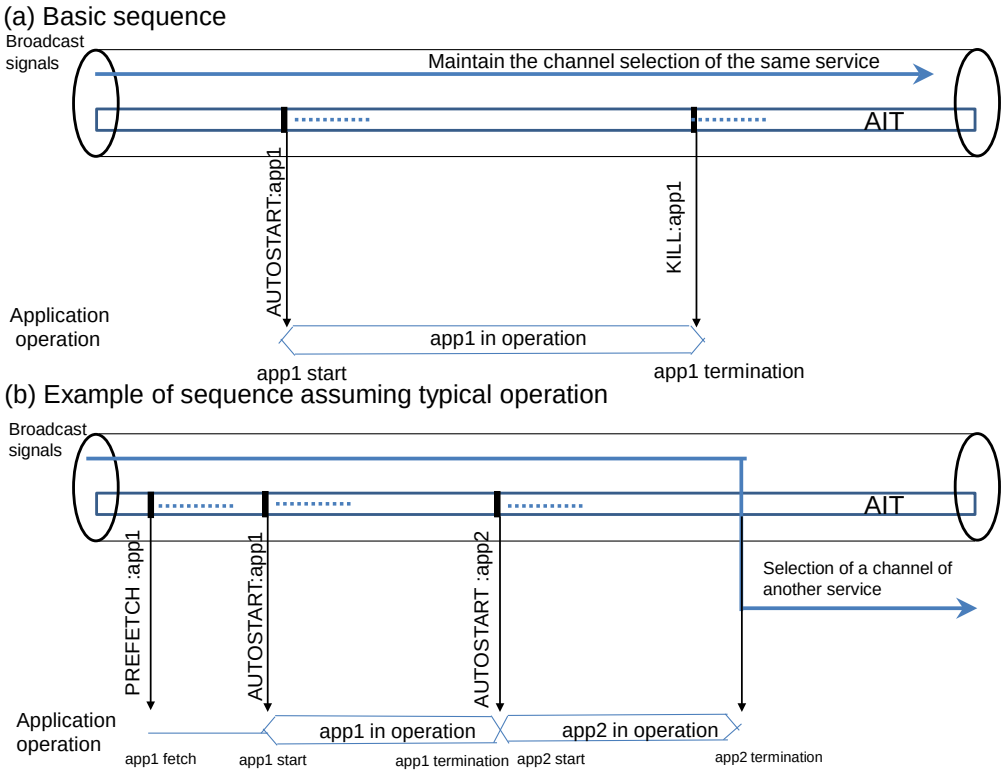


Figure 7-4 Lifecycle examples of broadcast-oriented managed applications in Scheme A

7.5.1.2. Scheme B: Control based on application control information referred to at the time of application launch

This scheme corresponds to the case where a URI in the AIT is specified when the application concerned is launched by another application or when it is launched by data broadcasting, as discussed in 7.5.2. Here, the case where the receiver fetches application control information transmitted in the data carousel by specifying arib-dc:// is assumed in addition to the case where the receiver fetches XML-format AIT or section-format AIT via communication as described in 7.1.3. In the former case, the application control information specifies only one application for launch, and the

application control code is limited to AUTOSTART. When the receiver is to launch an application, it fetches and refers to application control information, and then fetches and launches the application. After this, the receiver executes no further control that affects the application lifecycle. Therefore, the receiver continues to execute the application until the application makes a transition to another application or until it ends. Control over the application boundary continues during the application execution in accordance with the application control information fetched at the time of application launch. Figure 7-5 shows an operation example of this scheme. In this example, AIT is not transmitted via broadcast signals. Therefore, the receiver does not recognize the presence of the application, and so launches data broadcasting. When the receiver is instructed to launch an application by the startAITControlledApp() function, it refers to the URI of AIT, which is specified as an argument in the above function, fetches AIT from the AIT server in the communication network, and then launches app1 based on this AIT. While app1 is in operation, the receiver confirms the application boundary, etc., by referring to AIT, which it has already fetched. If the replaceApplication function is executed in app1, a transition is made to a new application (app2). This is done as follows. The receiver fetches AIT from the AIT server in the communication network by referring to the URI of AIT specified as an argument in the above function. The receiver launches app2 based on this. Later, app 2 terminates itself triggered by user operation.

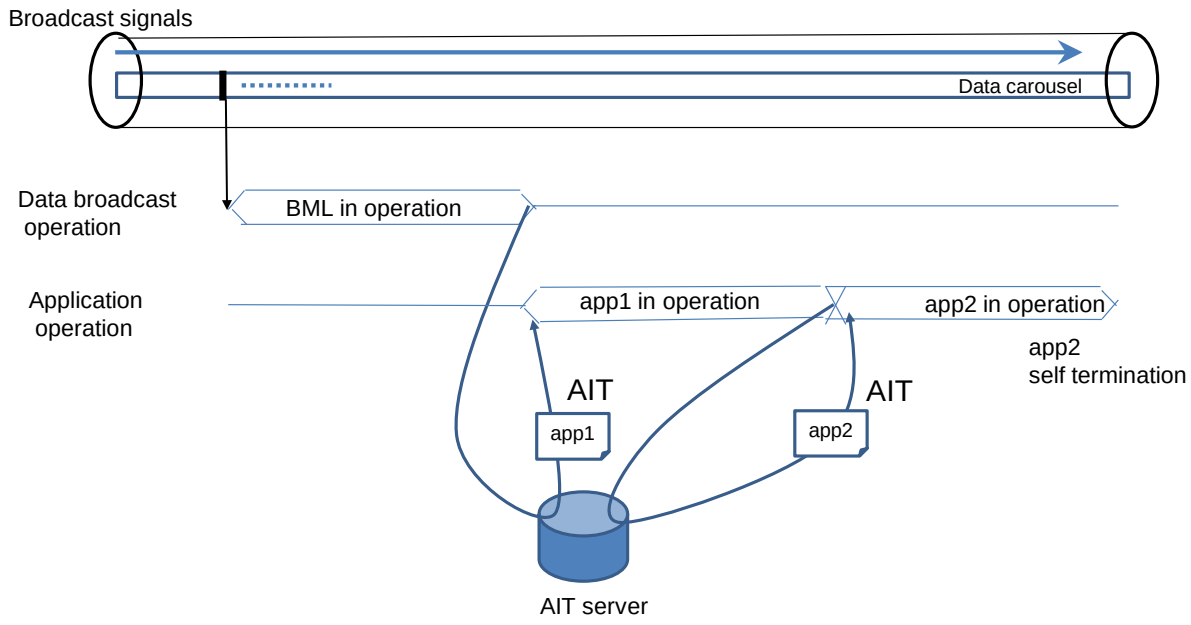


Figure 7-5 Lifecycle example of broadcast-oriented managed applications in Scheme B

7.5.2. Application fetch/launch

The receiver fetches applications in accordance with the method described in 7.4. The receiver fetches and then launches an application immediately except for the case where the application control code in the application control information indicates PREFETCH. In the latter case, the receiver fetches an application in advance. Only one application shall operate at a time. Therefore, only one application for which the application control code indicates AUTOSTART can be specified in the application control information. Three methods of fetching and launching an application are described below in relation to the lifecycle control schemes described in 7.5.1.

7.5.2.1. Fetch/launch based on application control information

This method is used when lifecycle control scheme A described in 7.5.1 is applied. When application control information is being transmitted via broadcast signals, the receiver monitors application control information and fetches this information at the time of channel selection or application control information update. If the application control code in AIT indicates PREFETCH, the receiver fetches an application from the specified application URL. As soon as the receiver fetches the application, it starts to manage and control the application. If the application control information code indicates AUTOSTART and the application's start priority is the highest, the receiver fetches the application if it has not fetched it yet, and launches it immediately.

7.5.2.2. Fetch/launch from another application

An application can launch another application. For this to happen, the receiver executes the replaceApplication function in an HTML document of the application. It refers to the application control information, and fetches and launches another application based on this information. The method of fetching application control information, and the method of controlling the application lifecycle are described below.

If ait_uri is not specified in an argument of the above function, the receiver operates in accordance with Scheme A described in 7.5.1. In other words, the receiver refers to the latest application control information by executing the above function. Only when the application specified by the application descriptor in the argument of the above function is AUTOSTART or PREFETCH, does the receiver fetch and launch the specified application. To refer to the latest application control information, the receiver must monitor application control information constantly.

On the other hand, if ait_uri is specified in an argument, the receiver operates in accordance with Scheme B described in 7.5.1. In other words, the receiver fetches and refers to application control information based on the URI of AIT specified at the time when the above function is executed. Only

when the application specified by the application descriptor in the argument of the above function is AUTOSTART, does the receiver fetch and launch the application.

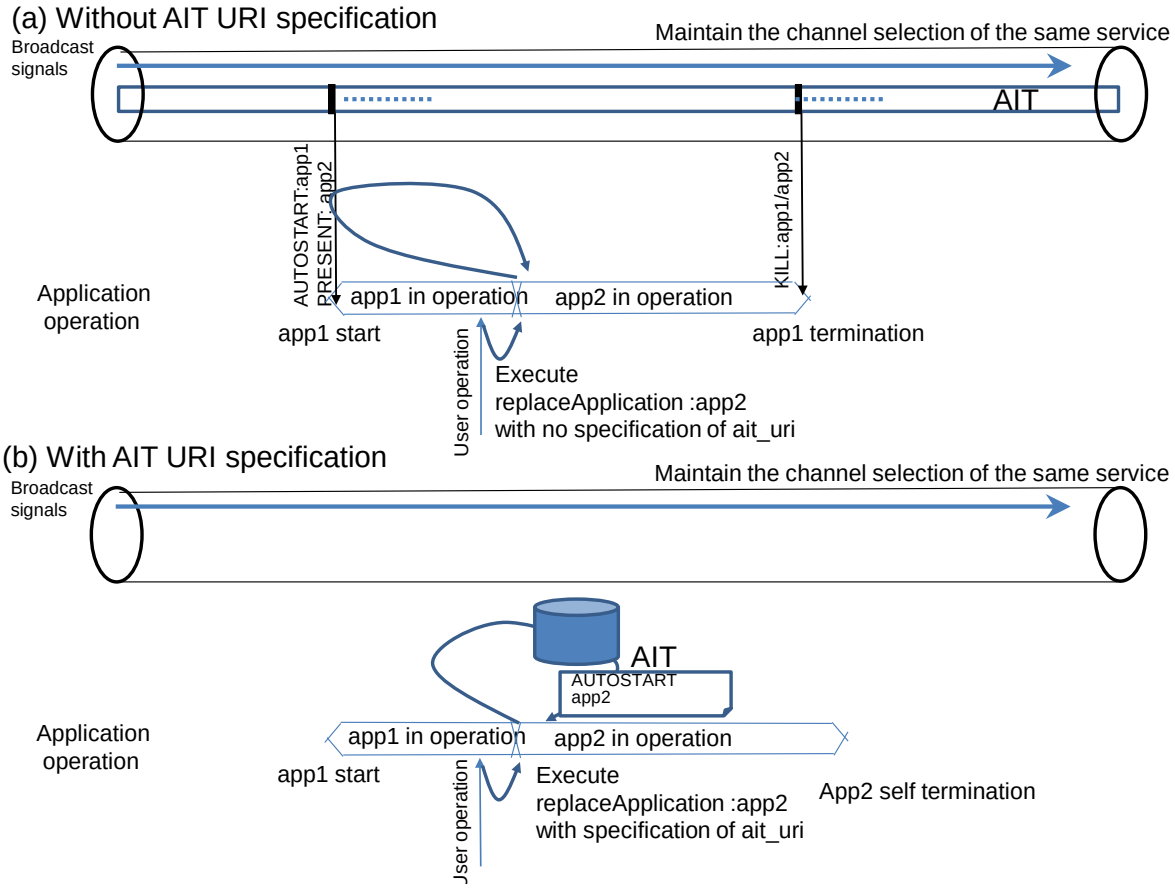


Figure 7-6 Launch of another application by an application

7.5.2.3. Fetch/launch from data broadcasting content

When an application is to be launched from data broadcasting content as a result of the execution of the startAITControlledApp() function, the receiver refers to application control information and fetches and launches an application in accordance with the description in the application control information. The application control information referred to here is fetched in the following manner.

If ait_uri is not specified in an argument of this function, the receiver operates in accordance with Scheme A described in 7.5.1. In other words, the receiver refers to the latest application control information by executing the above function. Only when the application specified by the application descriptor in the argument of the above function is AUTOSTART or PREFETCH, does the receiver fetch and launch the specified application. To refer to the latest application control information, the receiver must monitor application control information constantly.

On the other hand, if ait_uri is specified in an argument, the receiver operates in accordance with

Scheme B described in 7.5.1. In other words, the receiver fetches and refers to application control information based on the URI of AIT specified at the time when the above function is executed. Only when the application specified by the application descriptor in the argument of the above function is AUTOSTART, does the receiver fetch and launch the application.

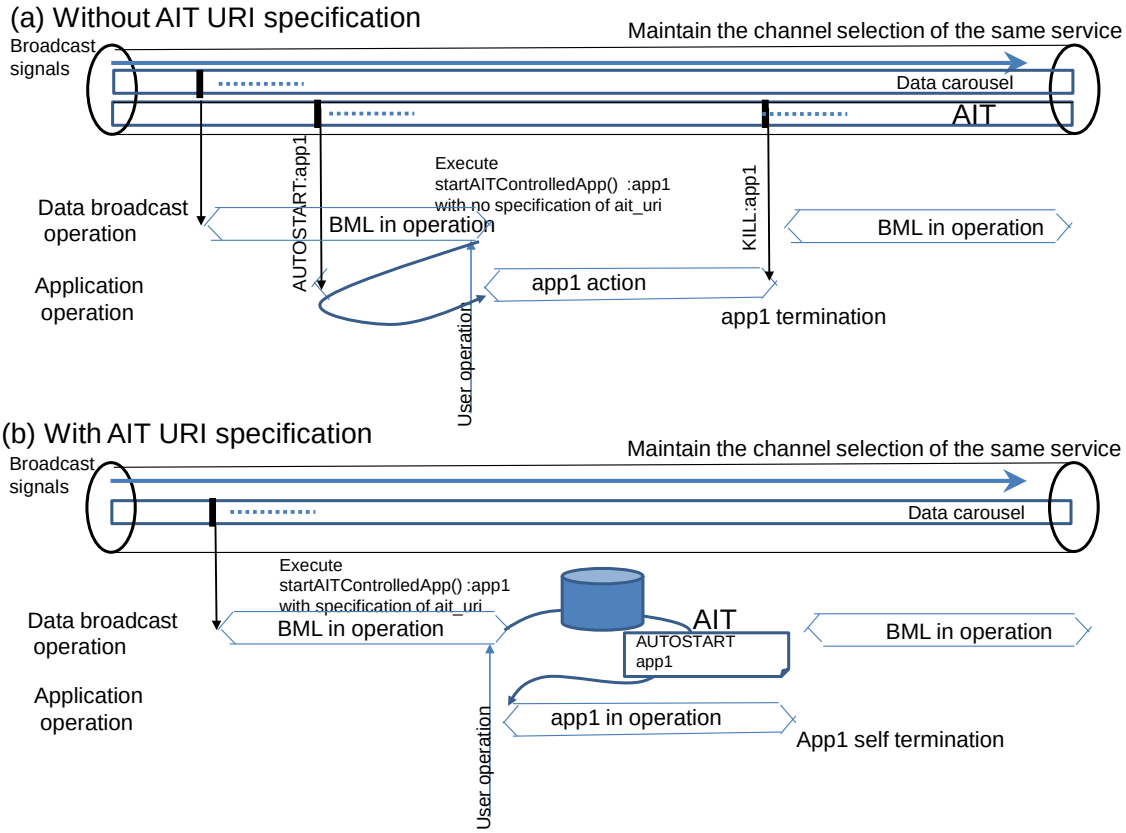


Figure 7-7 Application launch with data broadcasting

7.5.2.4. Fetch/launch of an application at the time of playing a recorded video

As described in 6.4.1, the launch of an application associated with a recorded video is equivalent to the launch of an application as a result of the user operation to “select a channel” that carries this content. Therefore, an application is launched based on the instruction of the AIT that the receiver has stored together with the content of the AIT that the receiver has fetched from the URI of the AIT. This launch operation follows the operations described in 7.5.2.1 and 7.5.2.3.

If the receiver fails to fetch the AIT or an application itself, it cannot launch the application. In such a case, the receiver continues to play the recorded audio/video.

7.5.3. Application termination

An application terminates in the events listed below. When an application terminates, its control is also terminated, and the application is removed from the scope of management. At the time of

application termination, the receiver re-fetches PMT, and re-determines the start priority, including the priority of data broadcasting.

1) Termination at the instruction of application control information

This termination method is available for lifecycle control based on Scheme A. If one of the following events occurs in cases where application control information is transmitted via broadcast signals, the application in operation is terminated.

- When application control information with its application control code indicating “KILL” is received
- When the reception of application control information indicating AUTOSTART or PRESENT for the application is discontinued. (The application is terminated due to timed-out.)
- When application control information for another application indicating AUTOSTART is received. (The original application is terminated, and the new application is launched.)

2) Termination by the application itself

An application terminates itself by calling a function that terminates the application.

3) Termination as a result of switching of the broadcast service

If the user switches the current broadcast service to another, the application is terminated.

4) Application transitions

When a transition is made from one broadcast-oriented managed application to another broadcast-oriented managed application or to an unmanaged application, etc., the application engine continues to operate, but the original application is deemed terminated.

7.6. Application boundary and broadcast resource access control

Broadcast-oriented managed applications are allowed access to broadcast resources, such as referring to broadcasting videos, etc., depending on the settings in the application control information. However, this assumes that the application is managed as a reliable application. Applications may make transitions from the entry document used at the time of application launch to a series of other documents. This can give rise to risks, such as transitions to documents outside the assumed scope of management and unauthorized access to broadcast resources. To prevent this, it is necessary to specify the domain in which transitions are allowed and to define the scope of application management so that only those documents within the scope of management are allowed access to broadcast resources. Even if a document is within the above scope of management, there are demands that it be made possible to set access permission to specific broadcast resources, domain by domain. It is also necessary to consider cases where broadcast-oriented managed applications refer to ordinary webpages using iframe elements. There are also demands that no HTML documents referred to using iframe elements be allowed access to broadcast resources at all, but there are also opposing demands that even such HTML documents be allowed access to some broadcast resources.

Furthermore, there are demands that some degree of freedom be given to transitions of HTML documents using iframe elements, and some opposing demands that, if there are links to outside a certain domain, such transitions be prohibited. Considering these demands, information about the settings of an application boundary and access permission in application control information is specified in ARIB STD-B24 Volume 4 Chapter 5 as an application boundary/permission descriptor.

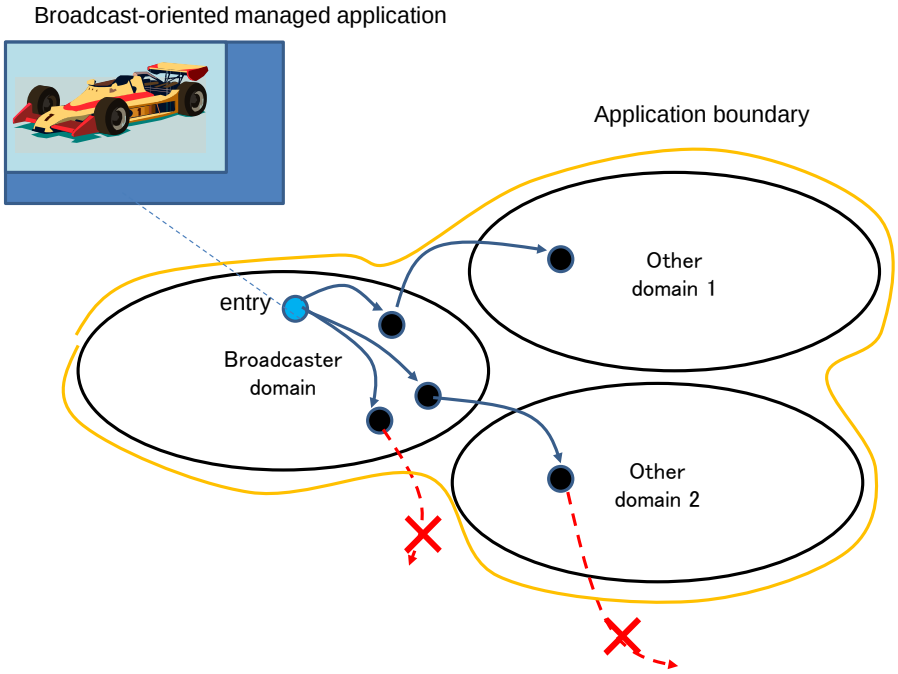
In these settings, an application boundary is defined as a domain in which transitions are allowed.

7.6.1. Application boundary setting

An application boundary is set in the application boundary/access permission setting information (application boundary/permission setting descriptor in AIT) in application control information. One or more specific Internet domains or their subdirectories are put together as a set of domains represented by URLs. As long as transitions are made to HTML documents fetched from within this domain, they are considered to be within the application boundary. Document transitions to outside the application boundary are prohibited, but how the receiver should behave when such transitions are instructed shall be specified in the operational rules. An application boundary can be changed by calling the setApplicationPermission function during the execution of the application.

The above rule applies not only to HTML documents that constitute broadcast-oriented managed applications, but also to HTML documents referred to using iframe elements in HTML documents of broadcast-oriented managed applications. The concept of application boundary is shown in Note: “x” indicates that this transition is prohibited.

Figure 7-8.



Note: “x” indicates that this transition is prohibited.

Figure 7-8 Concept of application boundaries

7.6.2. Individual control of access to broadcast resources

In the application boundary/access permission setting information (application boundary and permission setting descriptor in AIT) in application control information, an application boundary corresponds to a domain within which the broadcaster sets and manages access permission. Permission to access individual broadcast resources can be set domain by domain (URL). In addition, by calling the setApplicationPermission function, it is possible to change not only the application boundary but also access permissions, domain by domain (URL). The receiver recognizes individual broadcast resource access permissions set domain by domain and denies access if it is not permitted. If a certain location is simultaneously included in multiple domains specified by the URLs in the application boundary and permission descriptors or the setApplicationPermission function, and if conflicting permissions were set in these domains, the permission setting in the smallest domain takes precedence. If the domains are of the same size, the last specified permission setting takes precedence.

7.6.3. Application boundary at the time of playing a recorded video

As specified in 7.5.2.4, the execution of an application at the time of playing a recorded video is in accordance with the instructions of the AIT that are associated with the recorded video. Therefore, the application boundary is also determined based on these instructions.

7.7. Distribution of application caching and fetch attempts

Broadcast-oriented managed applications are required to work with broadcasting, which transmits data to multiple destinations. Therefore, all receivers are likely to fetch applications at the same time. This can cause a high load on the servers and the communication network concerned unless some measures are taken to avoid this situation. As such measures, methods of distributing application caching attempts in receivers and methods of distributing access attempts by receivers for fetching applications, etc., are specified below.

7.7.1. Application caching

7.7.1.1. Caching operation model

The following three measures are assumed for application caching.

(1) Precache

A measure to fetch and store an application before the application is to be executed in order to distribute the load of fetching applications and also to enable immediate launch of the application. This is handled by the application control part.

(2) Application caching that allows reuse

A measure to retain application resources after the first application execution if it is assumed that the application will be reused. This is handled by the application control part.

(3) Caching in the application engine (browser caching)

A measure for the application engine to retain each resource unit. Refer to 2.3.1 of IPTVFJ STD-0011 HTML5 Browser Specification for details.

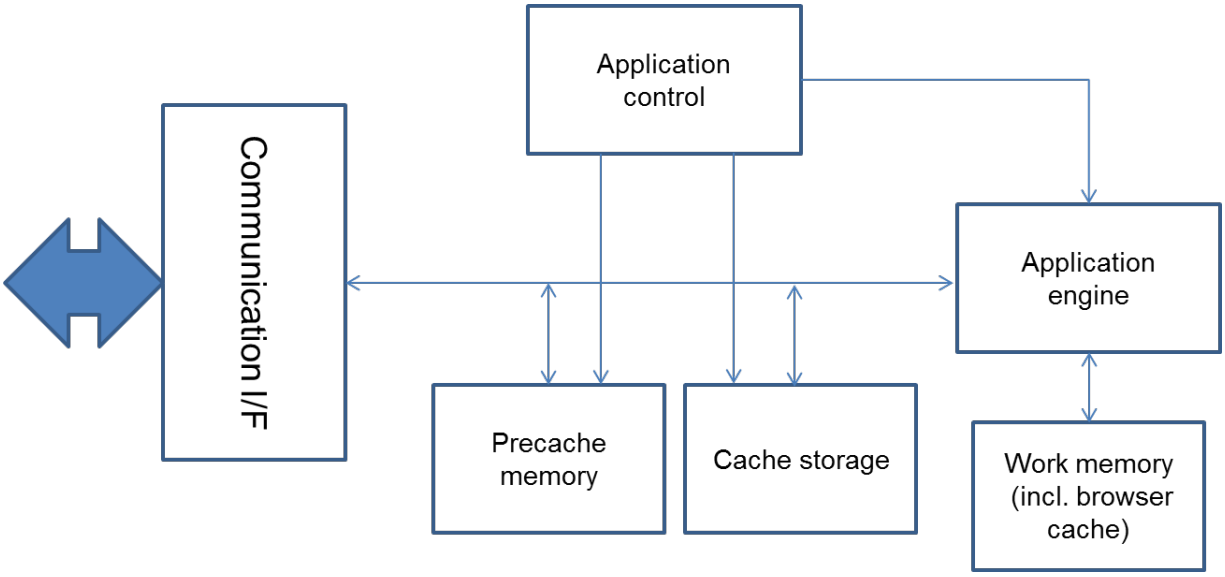


Figure 7-9 Receiver model involving application cache

7.7.1.2. Precaching

When the application control code in application control information indicates PREFETCH, the receiver fetches the relevant application file from the specified application URL, and stores it in the precache memory. This operation is not required if the file is already cached for reuse, as is described below. The application URL specifies only a file. If the application is not put together into a package, only the entry HTML document file is fetched and stored. The URL of the stored resource is also stored so that, when access to this URL is called for at the time of application execution, the required resource can be read from the precache memory and passed to the application engine.

7.7.1.3. Caching for reuse

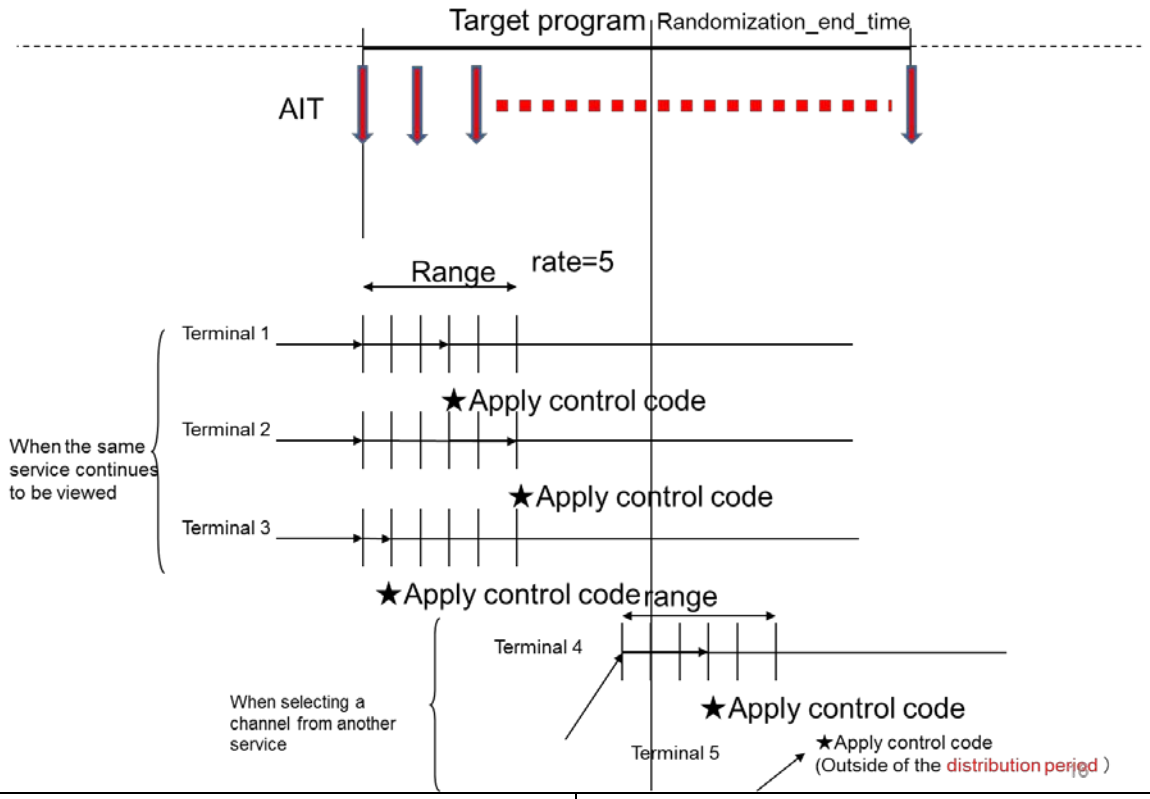
If cache control information (cache information descriptor in AIT) exists in the application control information and the application terminates, the receiver may store the application resource that is expected to be reused. If the receiver chooses to store the resource, it shall observe the instruction given in the cache control information. Whether to cache is up to the receiver. However, it is assumed that operational rules may make caching mandatory if the cache priority in the cache control information is high. As soon as the caching period in the cache control information has expired, the stored application resource shall be deleted. It may be deleted even before the expiration date. In addition, the associated application descriptors may also be stored so that, if there are instructions to fetch and execute an application based on application control information, the target application entry document can be read from the cache memory in accordance with the application descriptors referred to, and passed to the application engine. If the application version is specified in the cache control information and if the cache memory has the target application but its version is older than the

specified version, the application shall not be read from the cache memory but shall be fetched from the original source. Furthermore, what is stored in the cache memory shall be updated. Priority control information can be specified in the cache control information in cases where it is necessary to select only those applications that are no larger than the capacity of the cache memory. Details about application caching discussed in this section shall be specified in the operational rules.

7.7.2. Distribution of access attempts via a communication network

If a server access distribution parameter (stochastically applied delay descriptor in AIT) is included in application control information, the receiver shall stochastically delay the application of the application control code in accordance with the parameter setting. This is expected to slightly vary the timing of access by each receiver.

The receiver calculates delay time Td using a formula: Td= *k*×*range*÷*rate*, where *k* is a random integer between “0” and “*rate*.” It applies the application control code with a delay of Td from the AIT reception time. Since it is generally assumed that concentration of access attempts will be limited to the period around the start of the program, the period during which the above delay is applied can be set using Randomization_end_time.



現在の英語	以下のように修正して下さい
rate=5	Rate=5
range	Range

distribution period	distribution period

Figure 7-10 Server access distribution using a stochastically applied delay descriptor

Chapter 8. Communication Protocol Specification

8.1. Transmission of an application

This section specifies communication protocols and transmission type used by the receiver to fetch an application from the service provider's server.

8.1.1. Application transport protocol

The receiver uses the following protocols to fetch an application from the service provider's server.

- HTTP/HTTPS

8.1.1.1. HTTP/HTTPS

RFC2616 is assumed for the HTTP protocol. Unless indicated otherwise, the specification of this RFC shall be complied with. The SSL/TLS protocol shall comply with RFC2246, RFC4346 or RFC5246(TLS1.0, 1.1, 1.2). (Authentication of devices to check whether they support the integrated broadcast-broadband system will be specified separately.)

8.1.2. Application transmission method

The following methods are assumed for the receiver to fetch an application from the service provider's server.

- Fetch HTML files and the associated resources, file by file.
- Fetch a group of resources constituting an application as a single package

8.2. Data transport protocol between an application and the server

This section specifies the communication protocols used by the service provider's server and the receiver to exchange information for controlling applications. The data format at the application layer is service-dependent, and so is not specified here.

8.2.1. Transport protocol

The following communication protocols are used by the service provider's server and the receiver to exchange information for controlling applications.

- HTTP/HTTPS
- WebSocket

8.2.1.1. HTTP/HTTPS

RFC2616 is assumed for the HTTP protocol. Unless indicated otherwise, the specification of this RFC shall be complied with. The SSL/TLS protocol shall comply with RFC2246.

8.2.1.2. WebSocket

RFC6455 is assumed for the WebSocket protocol. Unless indicated otherwise, the specification of this RFC shall be complied with. However, this does not apply to certain fields that have been defined as an extension to this RFC.

8.3. Coordinated operation with companion devices

This Specification does not specify the mechanism a mobile information device and the receiver use to discover each other or the method they use to transmit control data in order to realize a service in which the application on the receiver connected to a home network and the application on the mobile information device work in a coordinated manner. Refer to Chapter 12 for the receiver operation related to working with companion devices.

Chapter 9. VOD

This chapter specifies how applications access VOD content.

9.1. Video stream transport protocol

This section specifies the video streaming protocol used to transmit to the receiver video streams that are accessed and displayed by applications. The following transport protocols are used:

- RTP/RTSP
- HTTP

9.1.1. Video transport protocol based on RTP/RTSP

Refer to IPTVFJ STD-0002 “IPTV Standard VOD Specifications Version 1.1” Section 4.1 “Video Transmission Protocol Based on RTP/RTSP” for the video transport protocol based on RTP/RTSP. Refer to IPTVFJ STD-0002 “IPTV Standard VOD Specifications Version 1.1” Section 6.5 “Transmission of Stream for Variable-speed Playback” for transmission of VOD content when variable-speed playback is implemented.

9.1.2. Video transport protocol based on HTTP

Refer to IPTVFJ STD-0002 “IPTV Standard VOD Specifications Version 1.1” Section 4.2 “Video Transmission Protocol Based on HTTP” for the video transmission protocol based on HTTP.

9.2. Video content

Video streams are multiplexed in one of the following ways.

9.2.1. IPTVFJ-TS method

This is specified in IPTVFJ STD-0002 “VOD Specifications Ver.1.1” 6.2.3 “Time-stamped TS.” The conditions specified in IPTVFJ STD-0002 “VOD Specifications Ver.1.1” 6.2.1 “Multiplexing within Service” and 6.2.2 “Details of Operation of MPEG-2 (system)” are applied to multiplexing within a stream.

9.2.2. MPEG-DASH method

This method uses “MP4 File Format (ISO/IEC 14496-12)” or “MPEG2-TS File Format (ISO/IEC 13818-1 and ITU-T Recommendation H.222.0), which is supported by ISO/IEC 23009-1 "Dynamic Adaptive streaming over HTTP (DASH) Part 1: Media presentation description and segment formats."

9.2.3. HLS method

This method uses MPEG2-TS File Format (ISO/IEC 13818-1 and ITU-T Recommendation H.222.0), which is supported by HLS (HTTP Live Streaming).

9.3. Content Playback Control Metafile

Refer to IPTVFJ STD-0002 “IPTV Standard VOD Specifications Version 1.1” Chapter 5 “Content Playback Control Metafile” for the transport protocol for a content playback control metafile and the format of a content playback control metafile that are applied in cases where IPTVFJ-TS is used for the multiplexing of video streams.

9.4. Digital rights management (DRM)

One of the following DRM methods is used, depending on the multiplexing method used.

9.4.1. In the case of the IPTVFJ-TS method

The DRM specification complies with IPTVFJ STD-0002 “IPTV Standard VOD Specifications Version 1.1” Chapter 7 “DRM Specifications.” In cases where Marlin IPTV-ES is used for digital rights management, refer to IPTVFJ STD-0002 “IPTV Standard VOD Specifications Version 1.1” Appendix B “Application Specifications of Marlin IPTV-ES System in DRM Specifications” for the DRM specification.

9.4.2. In the case of the MPEG-DASH method and the HLS method

A method that can be applied to EME (Encrypted Media Extensions) specified by W3C shall be used. Details shall be specified in the operational rules.

Chapter 10. Information Source Coding

This chapter specifies coding of the information source referred to by an application. It only specifies standard source coding schemes. It is not meant to limit the information sources that can be accessed by applications.

10.1. Video coding

10.1.1. MPEG-2 Video

The coding schemes specified in ARIB TR-B14 Volume 2 “Function Specification of Digital Terrestrial Television Broadcasting Receivers” and ARIB TR-B15 Volume 2 “Function Specification of BS Digital Receivers” shall be used for coding of MPEG-2 video.

10.1.2. H.264 | MPEG-4 AVC

The coding scheme specified in IPTVFJ STD-0002 “IPTV Standard VOD Specifications Version 1.1” shall be used for coding of H.264 video.

10.1.3. H.265 | ISO/IEC 23008-2 HEVC

This item is to be specified in the future.

10.2. Audio coding

10.2.1. MPEG-2 Audio

The coding schemes specified in ARIB TR-B14 Volume 2 “Function Specifications of Digital Terrestrial Television Broadcasting Receivers” and ARIB TR-B15 Volume 2 “Function Specification of BS Digital Receivers” shall be used for coding of MPEG-2 audio.

10.2.2. PCM (AIFF-C)

The coding schemes specified in ARIB TR-B14 Volume 2 “Function Specifications of Digital Terrestrial Television Broadcasting Receivers” and ARIB TR-B15 Volume 2 “Function Specification of BS Digital Receivers” shall be used for coding of PCM audio.

10.3. TTS

The coding scheme specified in IPTVFJ STD-0002 “IPTV Standard VOD Specifications Version 1.1” shall be used for coding for TTS (time-stamped TS) video.

10.4. Still images

10.4.1. JPEG

The coding scheme specified in ISO/IEC10918-1 shall be used for bitmap coding based on JPEG.

10.4.2. PNG

The format specified in ISO/IEC 15948:2004 shall be used for the file format of PNG graphics.

10.4.3. GIF

“Graphics Interchange Format Version 89a” specified by Compuserve Inc. of the U.S. shall be used for the file format of GIF graphics.

10.5. Characters

10.5.1. Character coding

UTF-8 specified in ISO/IEC10646:2012 shall be used for character coding.

10.5.2. Types of character code sets and code configuration

Of the coded character sets specified in ISO/IEC10646:2012, the font code points to be installed in the receiver shall be specified in the operational rules.

10.5.3. Character conversion

If broadcast resources are coded in schemes other than UTF-8, they shall be converted into UTF-8 in accordance with the correspondence specified in Appendix B of this Specification.

Chapter 11. Receiver Functions

11.1. Application engine

The application engine shall be an HTML5 browser based on IPTVFJ STD-0011 HTML5 Browser Specification.

11.2. Application control

11.2.1. Application control for broadcast-oriented managed applications

11.2.1.1. Fetch of application control information

The receiver fetches application control information, analyzes it and uses the analysis result to control and manage the application.

11.2.1.2. Application management

The application state transition model is shown in Figure 11-1. The following three application states are assumed:

- Enabled pre-cached: The application has been fetched. It can be launched.
- Enabled not pre-cached: The application has not been fetched yet. It can be launched.
- Active: The application is in operation.

The moment the receiver fetches application control information containing application control code that indicates “automatic start (AUTOSTART)”, “operable (PRESENT)”, or “prefetch (PREFETCH)” for the first time, for each application, it stores the necessary information and starts application management. Each application is identified by the organization identifier and the application identifier.

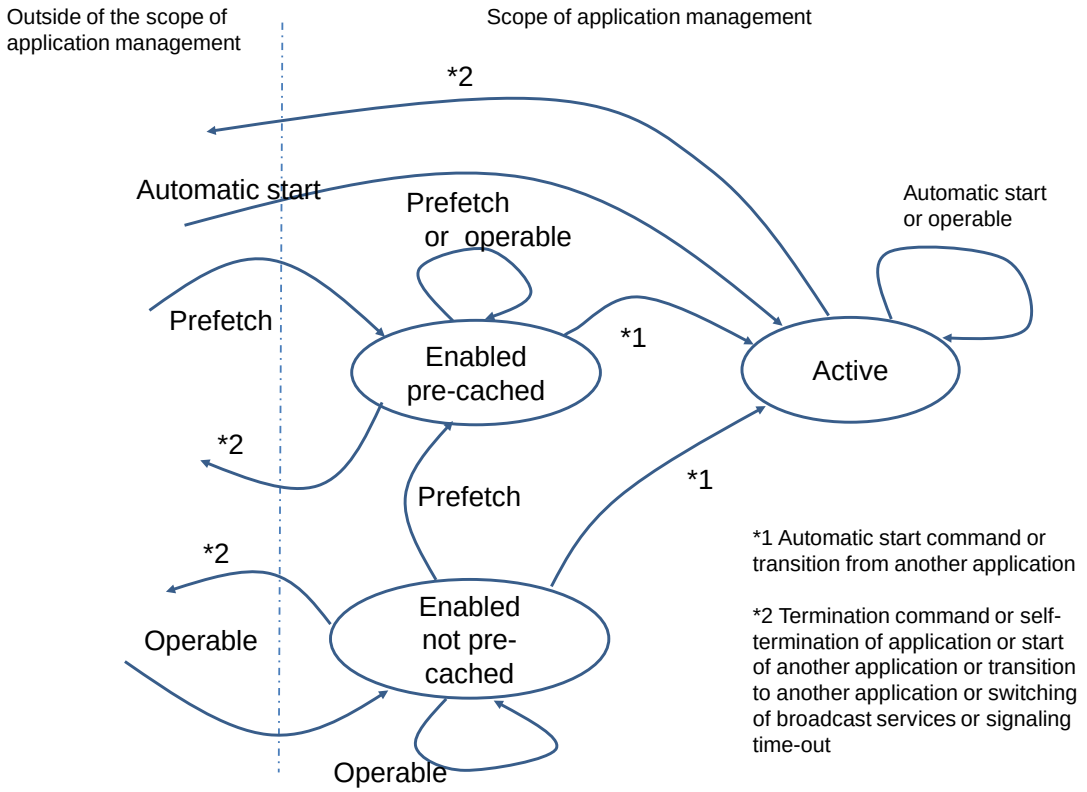


Figure 11-1 Application state transition model

After this, the receiver checks the organization identifier and the application indicator in the application control information fetched. If they match those of the application to be managed, the receiver executes the process specified in the application control code, and manages the resulting state transition. When the application terminates or makes a transition to another application, thus going outside the scope of management, the receiver terminates management of that application.

11.2.1.3. Application fetch process

When the receiver fetches application control information that indicates “PREFETCH,” it is expected to access the specified application URL, fetch the application and cache it. After this, when the receiver fetches application control information indicating AUTOSTART, the application engine launches the target application if it has been cached. Otherwise, the receiver accesses the specified application URL to fetch the application and passes it to the application engine. The application may be fetched via communication or via broadcast signals. In either case, the receiver needs to resolve the location based on the application control information and fetch the application appropriately.

11.2.1.4. Application launch process

When the receiver has fetched application control information indicating AUTOSTART at the time of the user's channel selection, of the start of a program, etc., it requests the application engine to launch the specified application provided that the application satisfies all of the following requirements:

- ✓ The application type and the application specification version specified in the application control information are operable ones (HTML5 in this Specification).
- ✓ If an application profile is included in the application control information, the receiver satisfies the specified operational requirements.
- ✓ If applications of different schemes, including data broadcasting and other application types, are set to AUTOSTART, the relevant operable application is given the highest start priority.

If an application is to be launched from data broadcasting or from another application, the receiver requests the application engine to launch the specified application provided that the specified application satisfies the following requirements:

- ✓ The application type and the application specification version specified in the application control information are operable ones (HTML5 in this Specification).
- ✓ Specified applications are in the "Enabled" state.
- ✓ If an application profile is included in the application control information, the receiver satisfies the specified operational requirements.

11.2.1.5. Application termination/transition

The control/management of the application in operation is terminated if the following application termination/transition occurs:

- ✓ The receiver has fetched application control information indicating KILL.
- ✓ In the case where the application engine can handle only one application at a time, the receiver has fetched application control information that orders automatic start of an application different from the one currently in operation.
- ✓ The user has selected another broadcast service, and the broadcast service has been switched as a result.
- ✓ The application has terminated itself in accordance with the application description due to user operation, etc.
- ✓ In the case where application control information is transmitted via broadcast signals, the receiver has not received application control information indicating PRESENT or AUTOSTART of a target application before the time-out occurs.
- ✓ The application has made a transition to another application in accordance with the application description due to user operation, etc.

When an application terminates and goes outside the scope of application management, resources associated with the application can be released. However, application termination does not necessarily result in the termination of the application engine. If a transition is made to another application, the application engine is obviously not terminated. Whether the application engine continues to operate when there is no application to be put into operation after an application terminates depends on the receiver implementation.

If an application terminates but no other application is launched at the same time, the receiver refers to PMT and operates in accordance with the start priority.

This is true even in cases where an application terminates itself in the middle of a program, etc., due to user operation, etc., in accordance with the application description. If an application complying with this Specification is selected due to its high priority and the receiver fetches application control information indicating automatic start of the application that terminated itself immediately before, the latter application is launched automatically.

11.2.1.6. Fetch of event messages

When the receiver receives an event message related to this system, it shall forward it to the application engine.

11.2.2. Application control for non-broadcast-oriented managed applications

Refer to 13.3 and 13.4.

11.3. Security management

11.3.1. Control of access by applications

Refer to 7.6 for control of access by broadcast-oriented managed applications. Access by non-broadcast-oriented managed applications is controlled as follows. As with broadcast-oriented managed application, ApplicationBoundary specified 14.6 shall be applied to control of access by the application itself. Authorization of access to broadcast resources shall be controlled by a combination of the BroadcastPermission element specified in 14.8 and control signals contained in the broadcast signals (Appendix F).

11.3.2. Control of presentation by applications

When the receiver detects an emergency warning broadcast (from an emergency warning system (EWS)) and an application that complies with this Specification is in operation, the receiver shall control the application presentation method appropriately, such as immediately switching to full-screen broadcast display. Details shall be specified in the operational rules. The same concept applies when new signals related to emergency warnings are to be specified in the future.

11.4. Broadcast reception

The receiver is assumed to have the broadcast reception functions specified in ARIB TR-B14 Volume 2 “Function Specification for Digital Terrestrial Television Broadcasting Receivers” and ARIB TR-B15 Volume 2 “Function Specification for BS digital Receivers.” (However, for the time being, refer to this Specification for reception functions that involve extensions to broadcast specifications and operational specifications.) Subtitles and superimposed subtitles from the received broadcast program shall always be displayed in the foreground, keeping any application, data broadcasting, or broadcast video in the background.

11.5. Playing of communication content

The communication content play function that the receiver shall have is a VOD streaming reception function based on the IPTVF STD-0002 VOD Specifications.

11.6. Control of subtitle data presentation and data fetch

It is assumed that managed applications access subtitles transmitted in a broadcast video based on the specification in ARIB STD-B24 Volume 1 Chapter 9.

Broadcast subtitles are presented by a receiver function. However, if an application is to control the presentation of broadcast subtitles or to fetch subtitle text, it uses an element that refers to the broadcast subtitle through an API that is used to fetch instructions on subtitle presentation or subtitle text.

Chapter 12. Collaboration with a Companion Device

12.1. System models for collaboration with companion devices

This section describes the system models for collaboration with a companion device assumed in this Specification. The system models can be broadly classified into a direct communication method, in which the receiver and the companion device concerned are connected to the same network and communicate with each other directly, and a server relay method, in which the receiver and the companion device concerned communicate with each other via a relay server in a network that is different from the ones to which they are connected. Each method includes a number of system models. All the system models are listed below:

- Direct communication method (Section 12.1.2)
  - System model A (Section 12.1.2.1)
  - System model B (Section 12.1.2.2)
  - System model A+B (Section 12.1.2.3)
- Server relay method (Section 12.1.3)
  - System model C (Section 12.1.3.1)
  - System model D (Section 12.1.3.2)

Note that these are just an exhaustive enumeration of the companion device collaboration models assumed in this Specification. This enumeration does not imply that companion device collaboration services provided based on this Specification must be able to support all these system models. It is mainly assumed that a system model and associated functions are selected in each instance of operation.

12.1.1. Reference model

A reference model that is common to all the system models is shown in Fig. 12-1. All the system models are specified based on this reference model

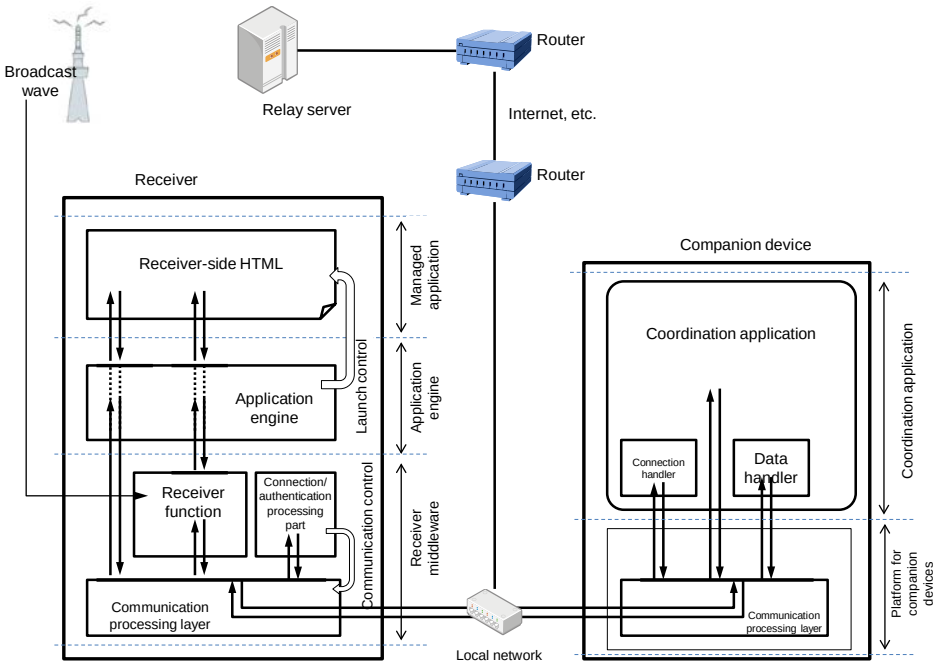


Fig. 12-1 Reference model for collaboration with a companion device

In this reference model, a companion device collaboration system consists of a receiver, a companion device, and a relay server used in the server relay method. The receiver consists of an application engine, managed applications that run on that engine, the connection/authentication processing part that is implemented as receiver middleware, a variety of receiver functions and the communication processing layer. The companion device consists of a platform on which the communication processing layer is implemented, and a collaboration application that runs on the platform. A connection handler and a data handler are built into the collaboration application. The connection handler discovers the service concerned and performs authentication for the receiver in the direct communication method. In the server relay method, it processes the establishment of a connection to a relay server, etc. The data handler receives data destined to the collaboration application from the receiver, or fetches this data and performs processing appropriate for the data in its capacity as a collaboration application.

Note that this figure is not intended to show how the collaboration application is implemented. The collaboration application can be implemented in a variety of ways. For example, it may be implemented as a native application, an application that can be directly executed on the platform of a companion device, as shown in Fig. 12-2. It may be implemented as a native application with a built-in connection handler and a built-in data handler but use HTML content as shown in Fig. 12-3. Alternatively, it may not use a special native application for collaboration with a companion device but be implemented in such a way that HTML content that runs on a browser also functions as the

connection handler and the data handler as shown in Fig. 12-4. In addition, it may be implemented as a composite of these. The reference model does not distinguish between these varieties of implementation.

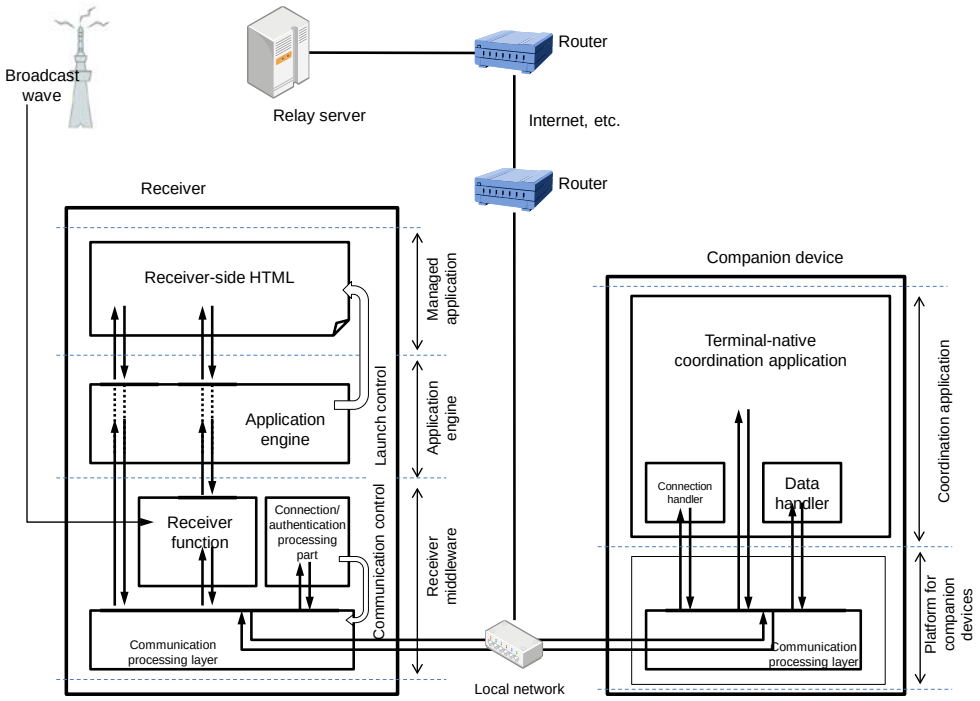


Fig. 12-2 Example of implementation of a collaboration application (single native application)

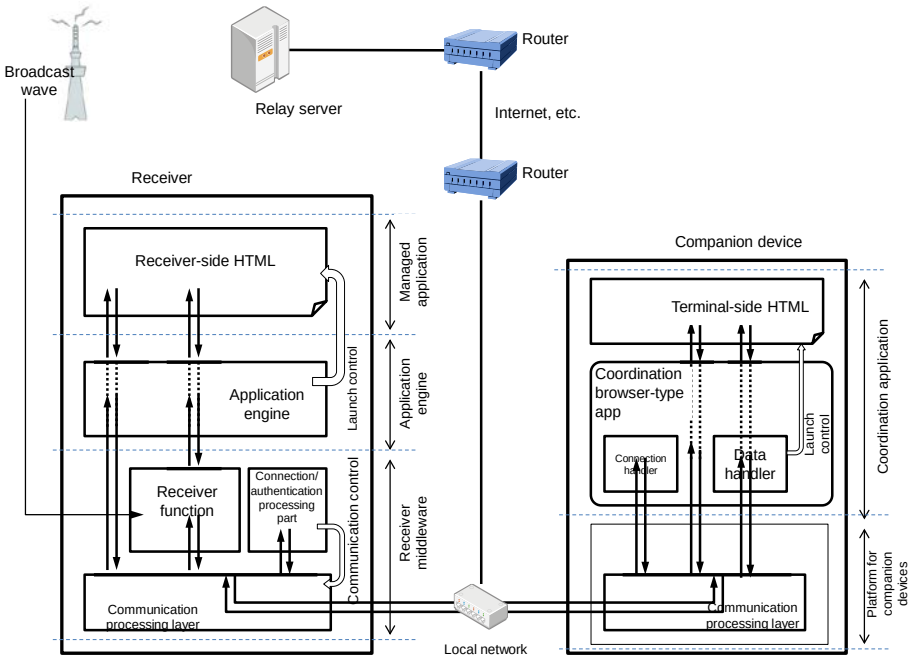


Fig. 12-3 Example of implementation of a collaboration application (browser-based custom application)

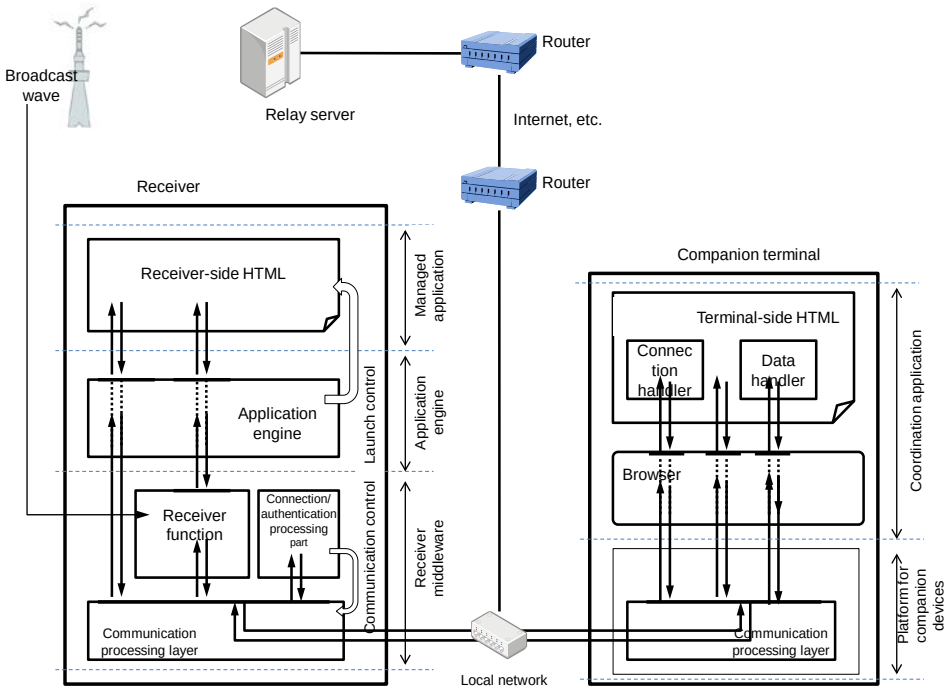


Fig. 12-4 Implementation example of a collaboration application (use of a browser)

12.1.2. Direct communication method

In the direction communication method, the receiver and a companion device are connected to the same network and communicate with each other directly. A network built within a home by the viewer (a so-called home network) is assumed for such a network.

12.1.2.1. System model A for collaboration with a companion device

In System model A, companion device collaboration is achieved through message communication between a managed application that runs on the receiver and a collaboration application.

The processing flow of System model A is shown below:

① Launch of the collaboration application
The user of a companion device installs a collaboration application and launches it (Fig. 12-5).

② Establishment of a connection between the receiver and a companion device
When the collaboration application is launched, the connection handler discovers the service concerned, and sends a connection request to the receiver to which the companion device is to be connected. The receiver that has received this request authenticates the requesting companion device. If it confirms the authenticity of both the companion device and the collaboration application, it accepts the connection request and establishes the requested connection (Fig. 12-6).
When the connection is established, the receiver stores information about the companion device to which it is connected (pairing) if necessary. Note that it is assumed that the above authentication involves judgment based on the range of operation requested by the collaboration application (access to receiver functions, etc.), user authentication, and negotiation regarding encryption of communication after the establishment of the connection.

③ Reception of a broadcast video and launch of a managed application
The receiver receives a broadcast video and launches the managed application concerned (Fig. 12-7). The broadcast video may contain data to be used by the collaboration application.

④ Transfer of data to be used by the collaboration application
When the receiver receives data for the collaboration application contained in the broadcast video, it stores the data. The managed application that runs on the receiver instructs the receiver to transfer the stored data to the companion device as necessary. The receiver does so, and the data handler of the collaboration application receives it (Fig.

12-8).

The content of the data for the collaboration application can vary depending on the service to be provided, the application, and the content. For example, it can be text or bitmap data to be displayed on the collaboration application, or an event, script, etc. that specifies the operation of the collaboration application. It can also be HTML content in the implementation example shown in Fig. 12-3 and Fig. 12-4. The processing flow for a case where HTML content is transferred as data for the collaboration application is shown in Fig. 12-9 and Fig. 12-10. Similarly, the receiver may transfer a URL instead of HTML content. In this case, the data handler that has received it fetches HTML content from the given URL and launches it.

⑤ Access to receiver functions by the collaboration application

The collaboration application accesses receiver functions via a managed application that runs on the receiver. In this model, the API used to access receiver functions is disclosed only to managed applications on the receiver. When the collaboration application wants to access receiver functions, it sends an access request using message communication between the collaboration application and the managed application. When the managed application receives the request, it operates the API, and sends the result to the collaboration application if necessary (Fig. 12-11).

⑥ Thereafter, the managed application on the receiver and the collaboration application execute their respective operations and also ④ and ⑤ as appropriate to achieve companion device collaboration.

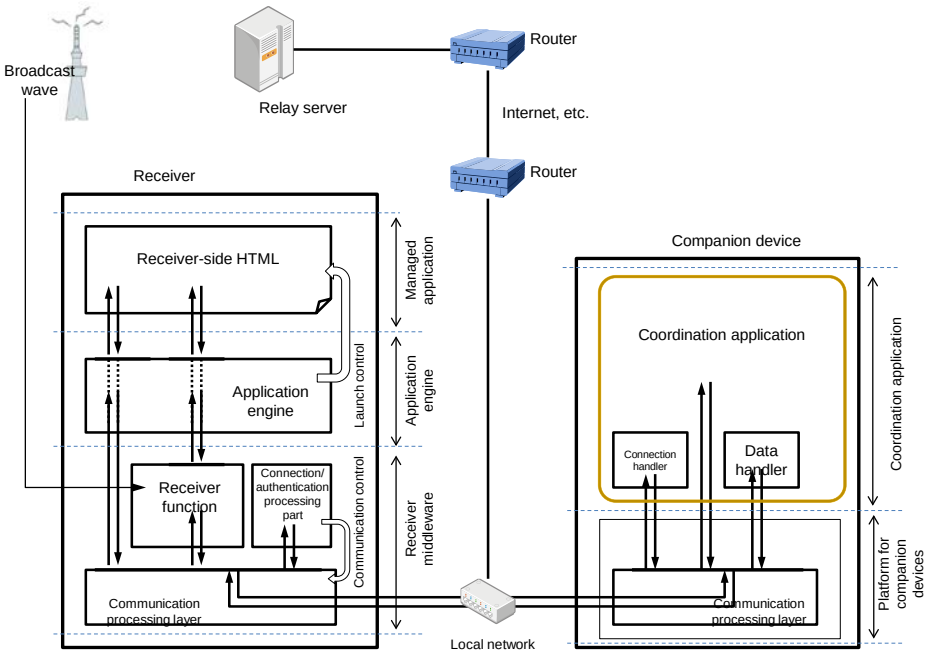


Fig. 12-5 Processing flow of System model A for collaboration with a companion device ①

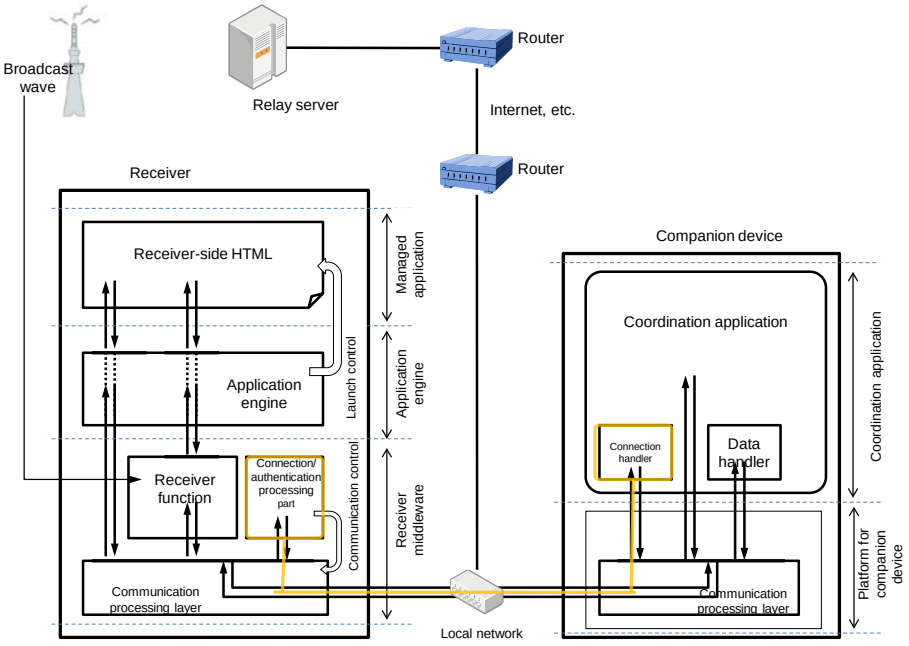


Fig. 12-6 Processing flow of System model A for collaboration with a companion device ②

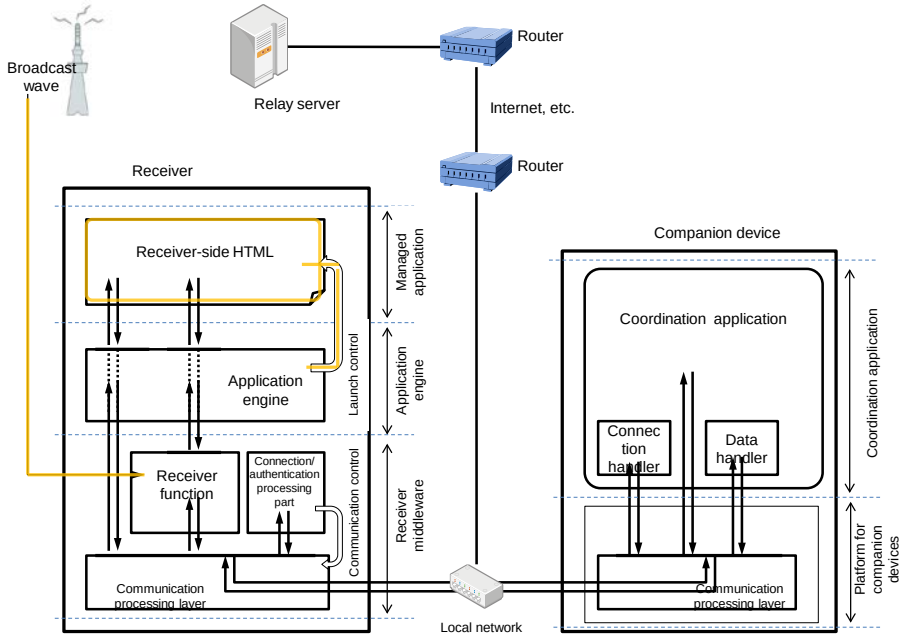


Fig. 12-7 Processing flow of System model A for collaboration with a companion device ③

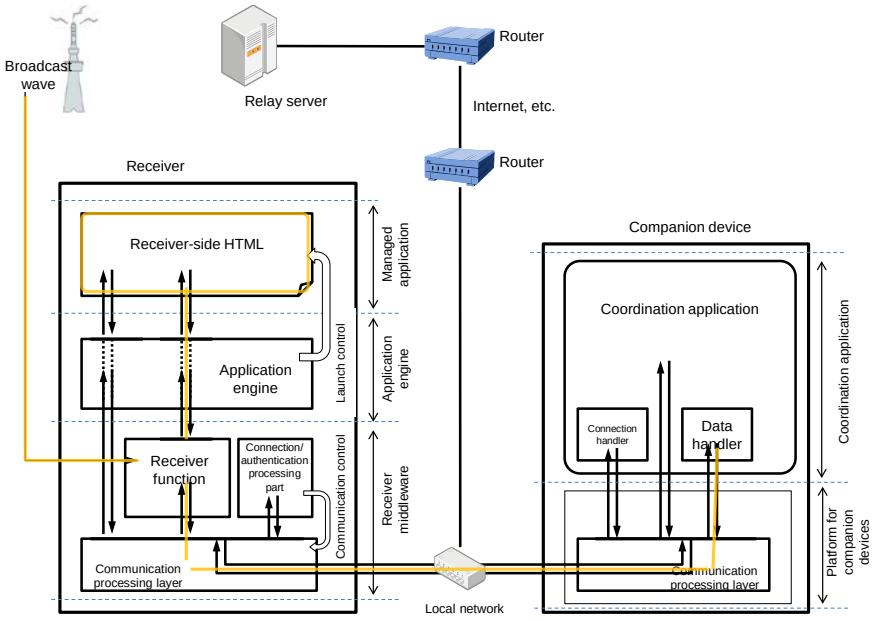


Fig. 12-8 Processing flow of System model A for collaboration with a companion device ④

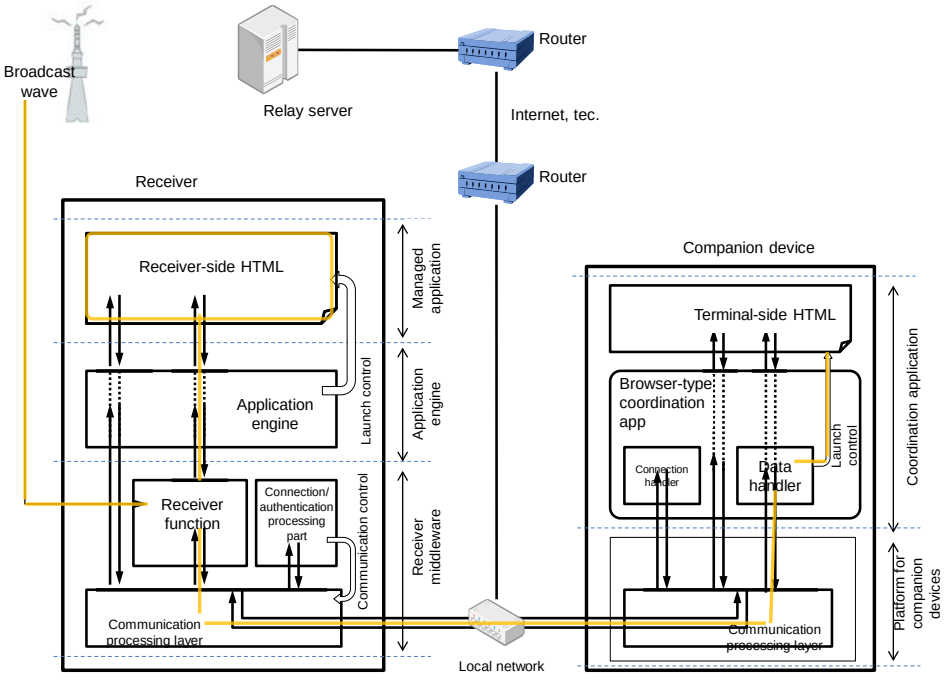


Fig. 12-9 Processing flow of System model A for collaboration with a companion device ④ (details for a browser-type application)

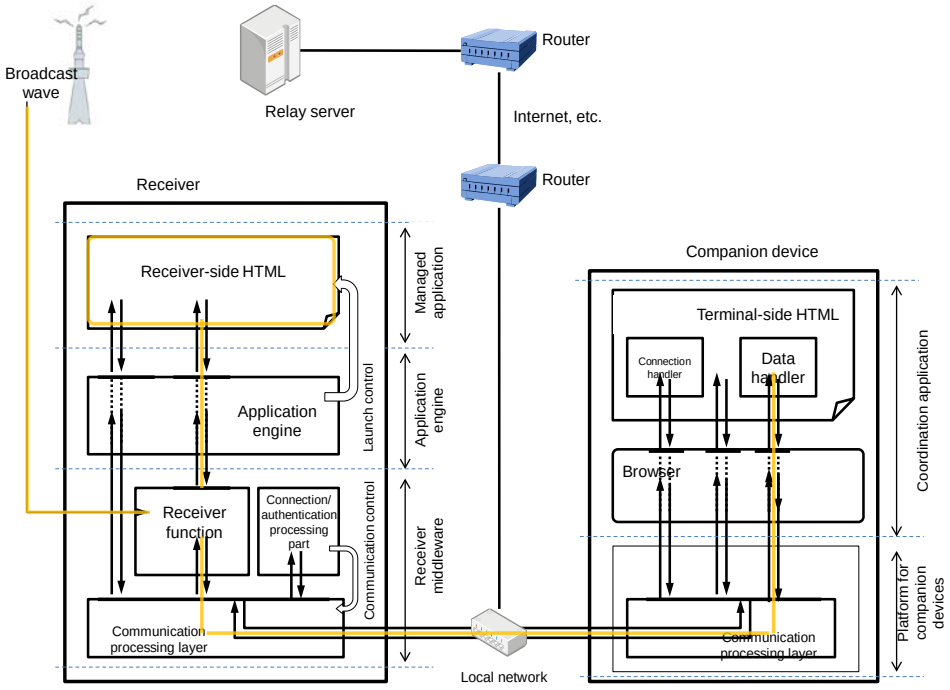


Fig. 12-10 Processing flow of System model A for collaboration with a companion device ④ (details for a browser-type application)

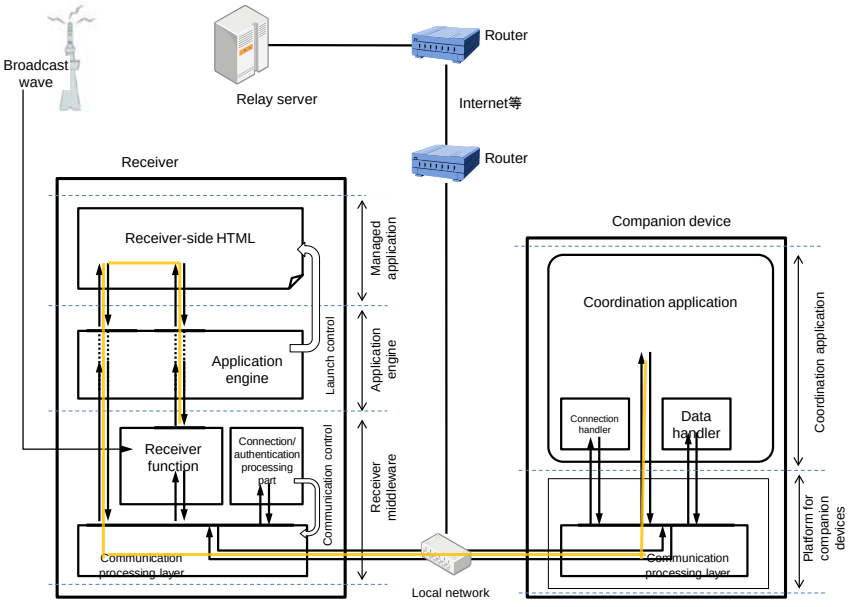


Fig. 12-11 Processing flow of System model A for collaboration with a companion device ⑤

12.1.2.2. System model B for collaboration with a companion device

In System model B, a function built into the receiver communicates with the collaboration application to achieve companion device collaboration through control by broadcast signals. The processing flow of System model B is shown below:

① Launch of the collaboration application
The user of a companion device installs a collaboration application and launches it (Fig. 12-12).

② Establishment of a connection between the receiver and a companion device
When the collaboration application is launched, the connection handler discovers the service concerned, and sends a connection request to the receiver to which the companion device is to be connected. The receiver that has received this request authenticates the requesting companion device. If it confirms the authenticity of both the companion device and the collaboration application, it accepts the connection request and establishes the requested connection (Fig. 12-6). When the connection is established, the receiver stores information about the companion device to which it is connected (pairing) if necessary.
Note that it is assumed that the above authentication involves judgment based on the range of operation requested by the collaboration application (access to receiver functions, etc.), user authentication, and negotiation regarding encryption of communication after the establishment of the connection.

③ Reception of a broadcast video
The receiver receives a broadcast video. The broadcast video may contain data to be used by the collaboration application and an instruction for data to be transferred to the collaboration application (Fig. 12-14).

④ Transfer of data to be used by the collaboration application
When the receiver receives data for the collaboration application contained in the broadcast video, it stores the data. If it receives an instruction to transfer data for the collaboration application to use, it transfers the given data to the specified companion device, and the data handler of the collaboration application receives it. Alternatively, the collaboration application polls the receiver to check for the presence of data. If data is present, the collaboration application obtains it from the receiver (Fig. 12-15).
The content of the data for the collaboration application can vary depending on the service to be provided, the application, and the content. For example, it can be text or bitmap data to be displayed on the collaboration application, or an event, script, etc. that specifies the operation of the collaboration application. It can also be HTML content in then implementation example shown in Fig. 12-3 and Fig. 12-4. The processing flow for a case where HTML content is transferred as data for the collaboration application is shown in Fig.

12-9 and Fig. 12-10. Similarly, the receiver may transfer a URL instead of HTML content. In this case, the data handler that has received it fetches HTML content from the given URL and launches it.

The format of the instruction to transfer data for the collaboration application to use also varies depending on the service to be provided, the application, and the content. Therefore, the data may also serve as an instruction to transfer data. For example, there can be an implementation in which reception of an event received as data for the collaboration application may be interpreted as an instruction to transfer data.

⑤ Access to receiver functions by the collaboration application

In this model, the API used to access receiver functions is disclosed to the collaboration application as appropriate, and the collaboration application accesses receiver functions via this API (Fig. 12-18).

⑥ Thereafter, the receiver and the collaboration application executes their respective operations and also ④ and ⑤ as appropriate to achieve companion device collaboration.

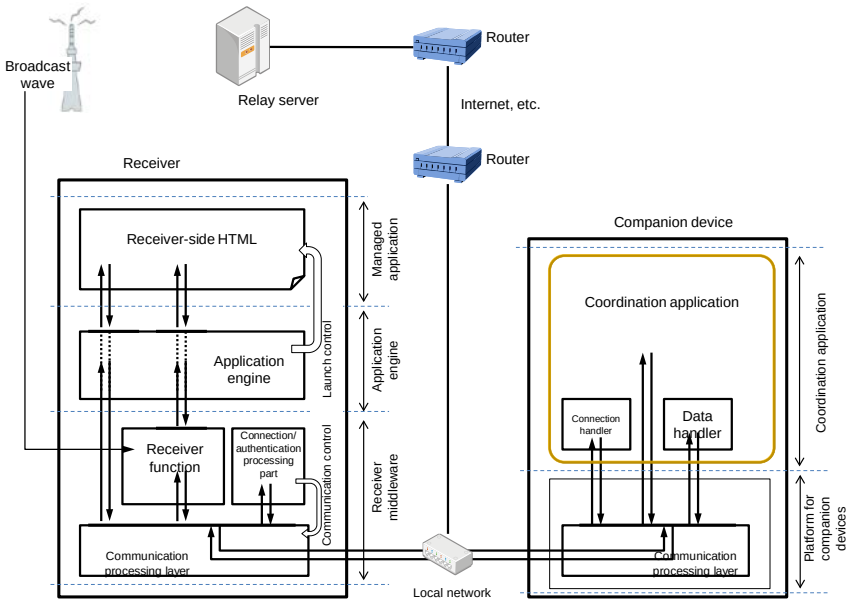


Fig. 12-12 Processing flow of System model B for collaboration with a companion device ①

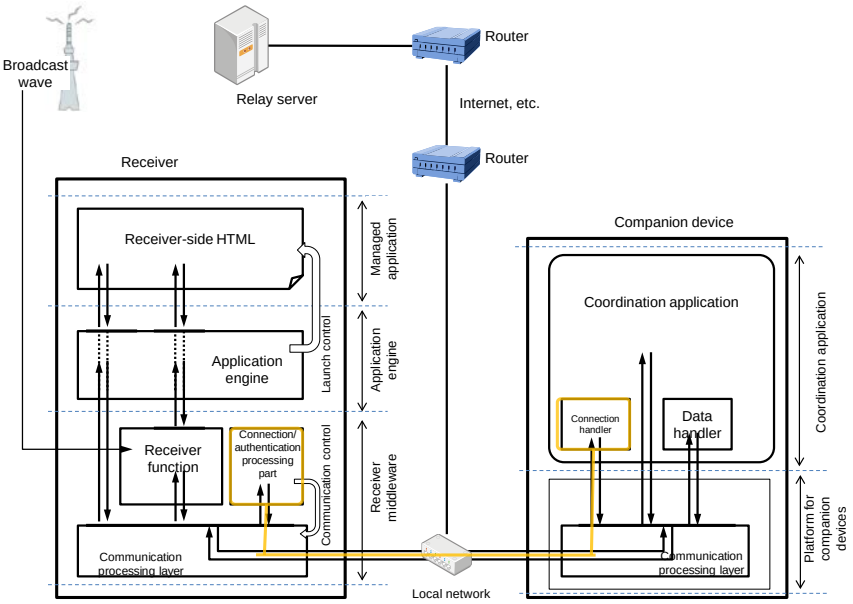


Fig. 12-13 Processing flow of System model B for collaboration with a companion device ②

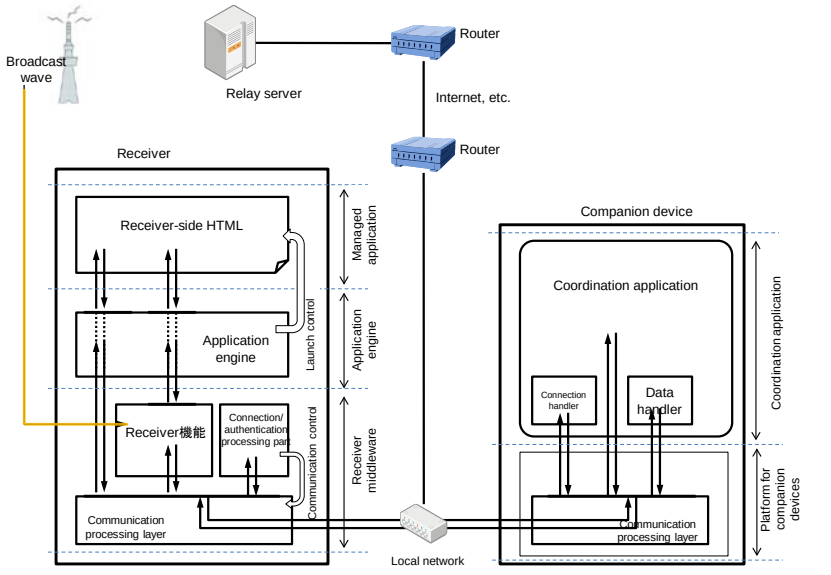


Fig. 12-14 Processing flow of System model B for collaboration with a companion device ③

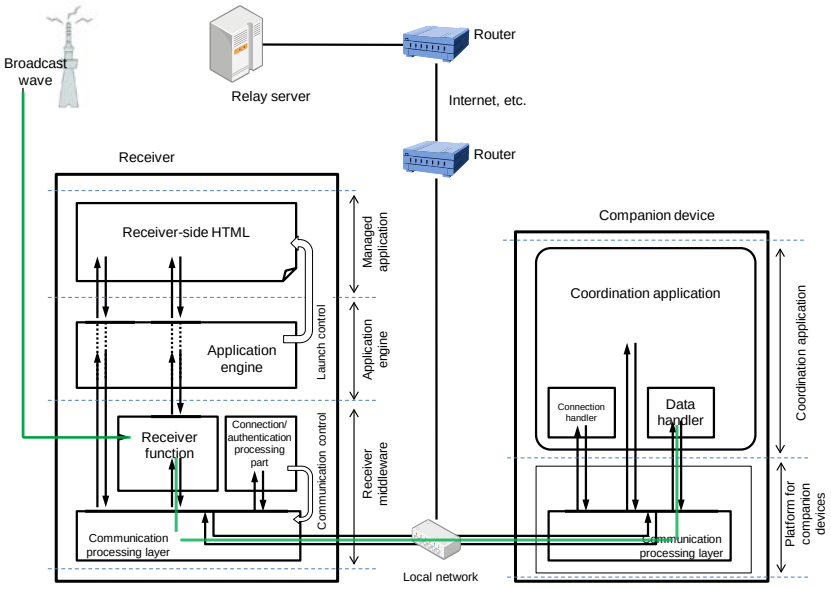


Fig. 12-15 Processing flow of System model B for collaboration with a companion device ④

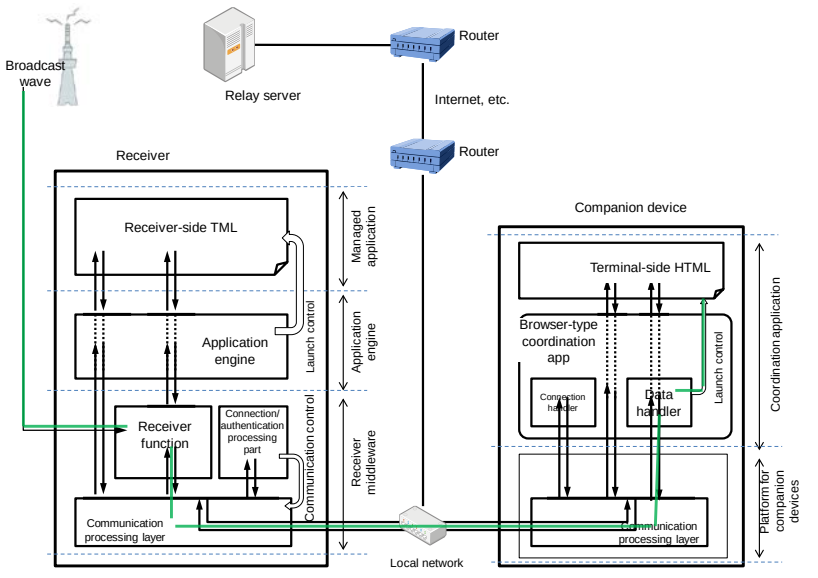


Fig. 12-16 Processing flow of System model B for collaboration with a companion device ④ (details for a browser-type application)

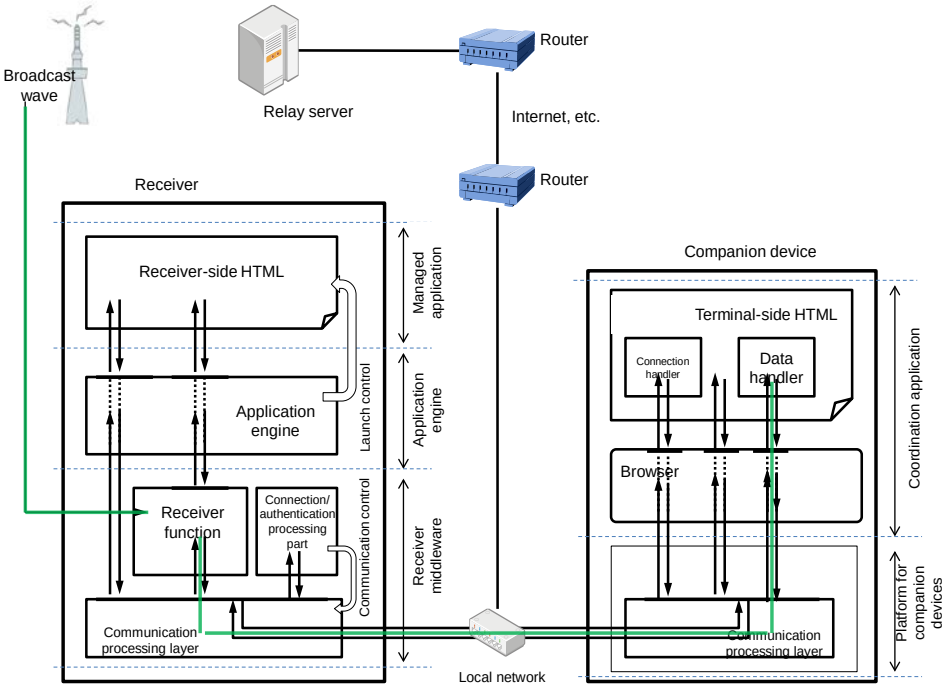


Fig. 12-17 Processing flow of System model B for collaboration with a companion device ④ (details for a browser-type application)

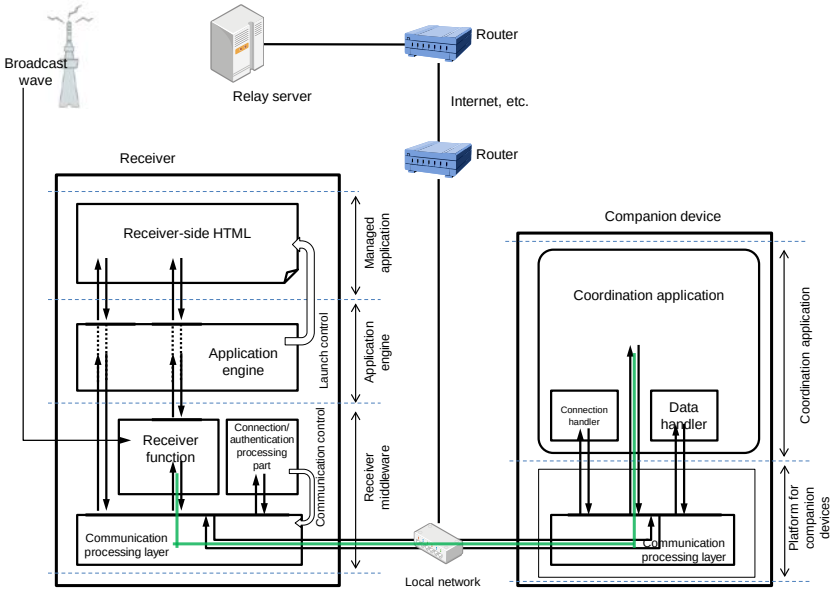


Fig. 12-18 Processing flow of System model B for collaboration with a companion device ⑤

12.1.2.3. System model A+B companion for device collaboration system model A+B

System model A+B uses both System model A and System model B. The processing flow of System

model A+B is shown below:

① Preparation of the collaboration application
The user of a companion device installs a collaboration application and launches it (Fig. 12-19).

② Launch of the collaboration application and establishment of a connection between the receiver and a companion device
When the collaboration application is launched, the connection handler discovers the service concerned, and sends a connection request to the receiver to which the companion device is to be connected. The receiver that has received this request authenticates the requesting companion device. If it confirms the authenticity of both the companion device and the collaboration application, it accepts the connection request and establishes the requested connection (Fig. 12-6). When the connection is established, the receiver stores information about the companion device to which it is connected (pairing) if necessary.
Note that it is assumed that the above authentication involves judgment based on the range of operation requested by the collaboration application (access to receiver functions, etc.), user authentication, and negotiation regarding encryption of communication after the establishment of the connection.

③ Reception of a broadcast video and launch of a managed application
The receiver receives a broadcast video. The broadcast video may contain data to be used by the collaboration application and an instruction to transfer the data (Fig. 12-21).

④ Transfer of data to be used by the collaboration application
When the receiver receives data for the collaboration application contained in the broadcast video, it stores the data. If the receiver function receives an instruction to transfer data for the collaboration application or if a managed application that runs on the receiver instructs the receiver to transfer the stored data, it transfers the given data to the specified companion device, and the data handler of the collaboration application receives it. Alternatively, the collaboration application polls the receiver to check for the presence of data. If data is present, the collaboration application obtains it from the receiver (Fig. 12-22).
The content of the data for the collaboration application can vary depending on the service to be provided, the application, and the content. For example, it can be text or bitmap data to be displayed on the collaboration application, or an event, script, etc. that specifies the operation of the collaboration application. It can be HTML content in then implementation example shown in Fig. 12-3 and Fig. 12-4. The processing flow for a case where HTML content is transferred as data for the collaboration application is shown in Fig. 12-9 and Fig. 12-10. Similarly, the receiver may transfer an URL instead of HTML content. In that

case, the data handler that has received it fetches HTML content from the given URL and launches it.

⑤ Access to receiver functions by the collaboration application

In this model, the collaboration application accesses receiver functions via the API disclosed to it. Alternatively, it sends an access request using message communication between the collaboration application and the managed application. When the managed application receives the request, it operates the API, and sends the result to the collaboration application if necessary (Fig. 12-25).

⑥ Thereafter, the receiver, the managed application on the receiver and the collaboration application execute their respective operations and ④ and ⑤ as appropriate to achieve companion device collaboration.

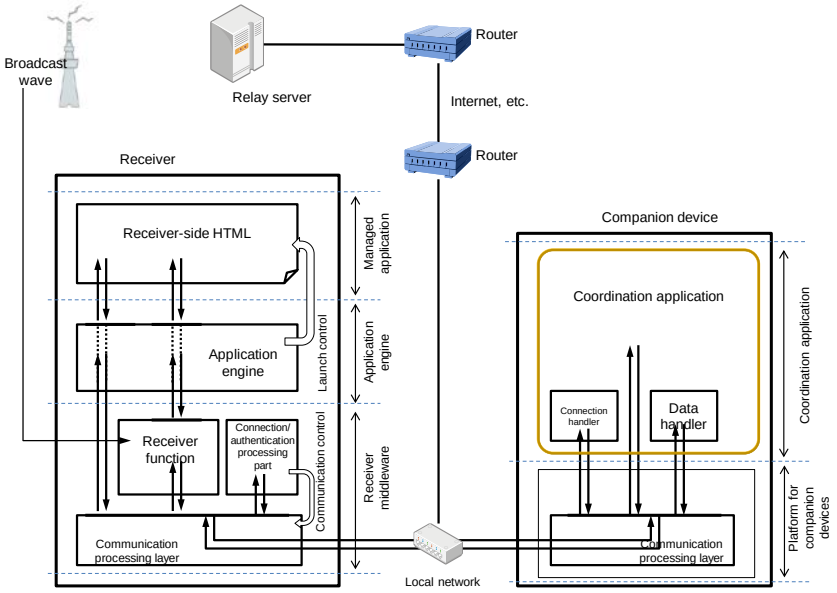


Fig. 12-19 Processing flow of System model A+B for collaboration with a companion device ①

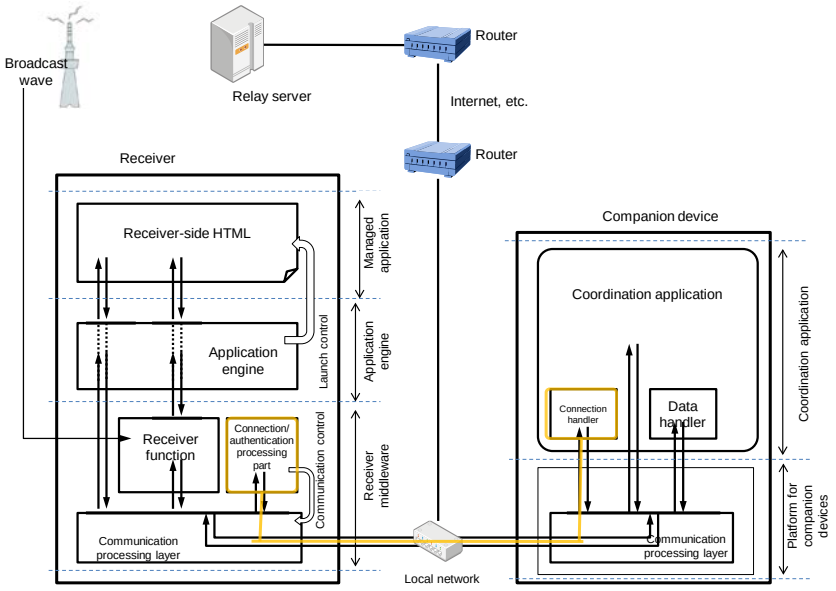


Fig. 12-20 Processing flow of System model A+B for collaboration with a companion device ②

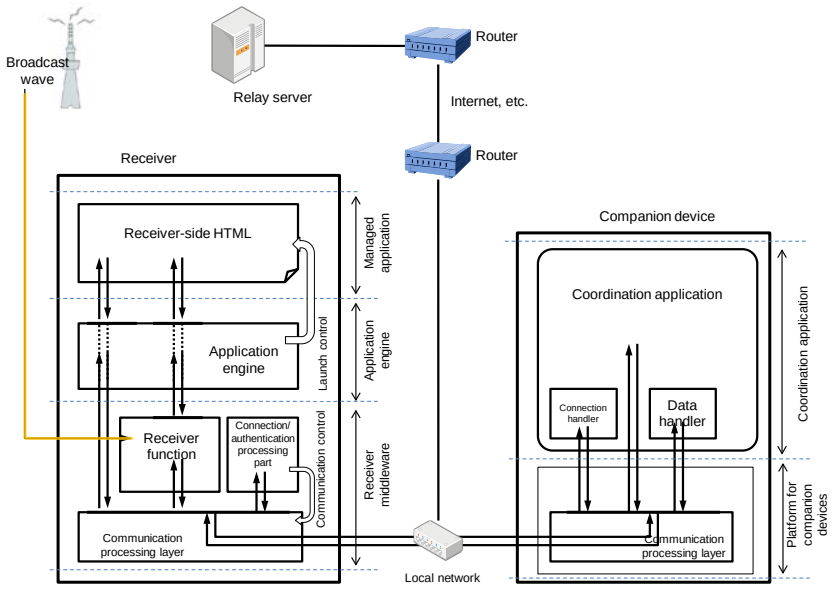


Fig. 12-21 Processing flow of System model A+B for collaboration with a companion device ③

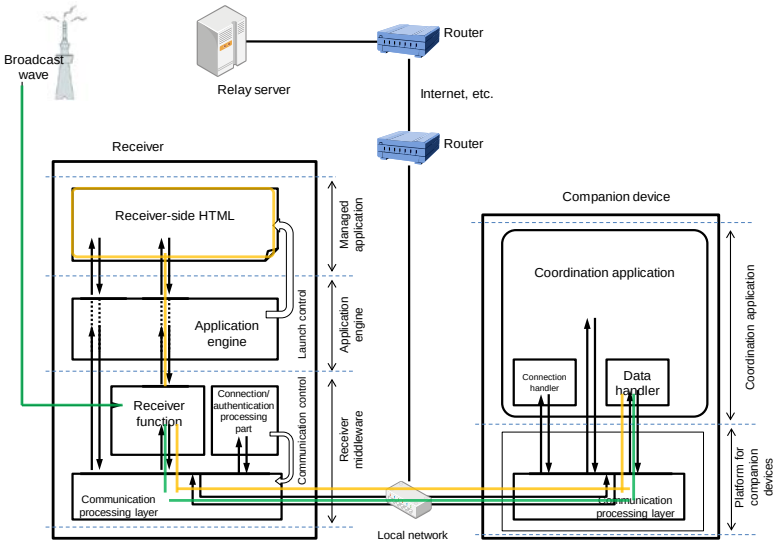


Fig. 12-22 Processing flow of System model A+B for collaboration with a companion device ④

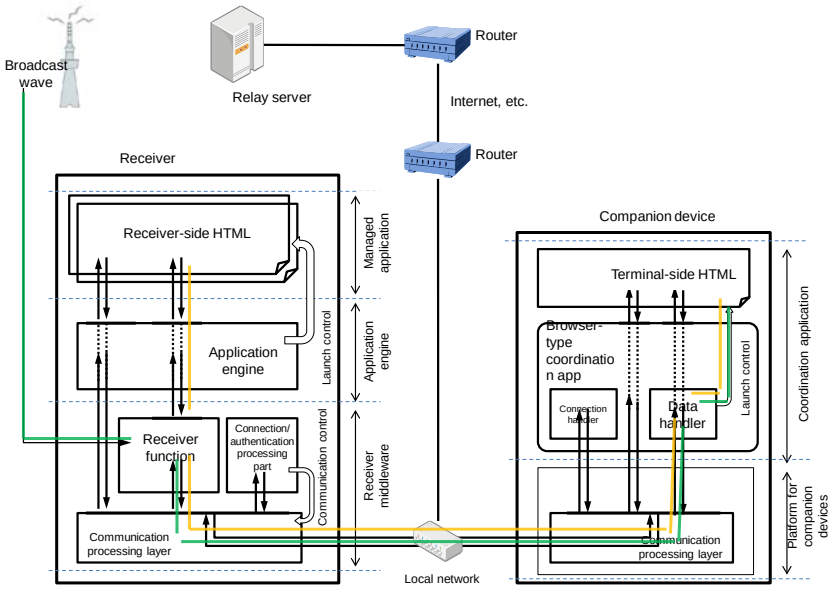


Fig. 12-23 Processing flow of System model A+B for collaboration with a companion device ④ (details for a browser-type application)

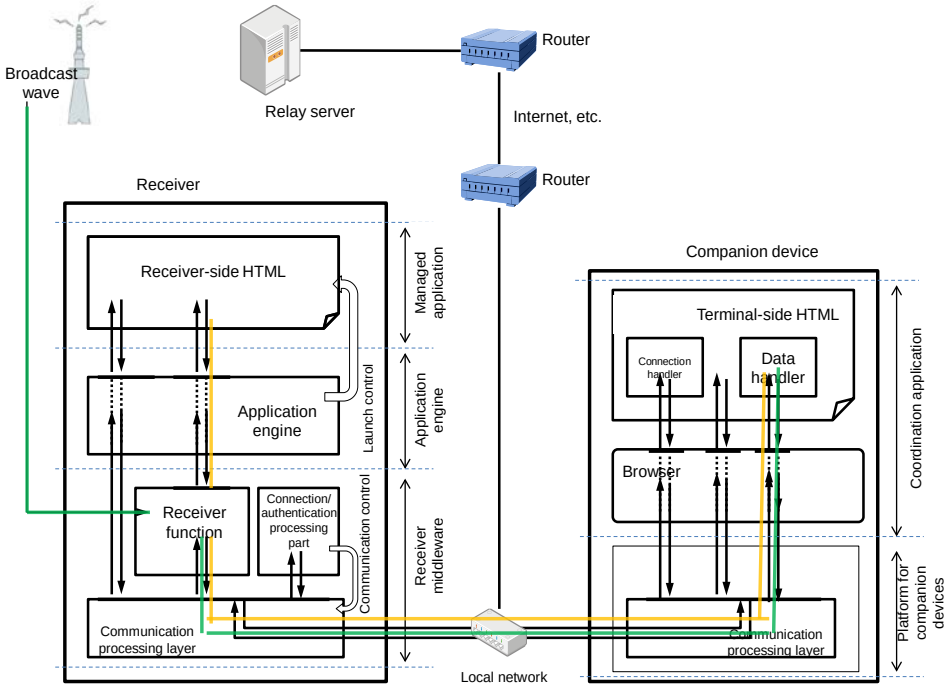


Fig. 12-24 Processing flow of System model A+B for collaboration with a companion device ④ (details for a browser-type application)

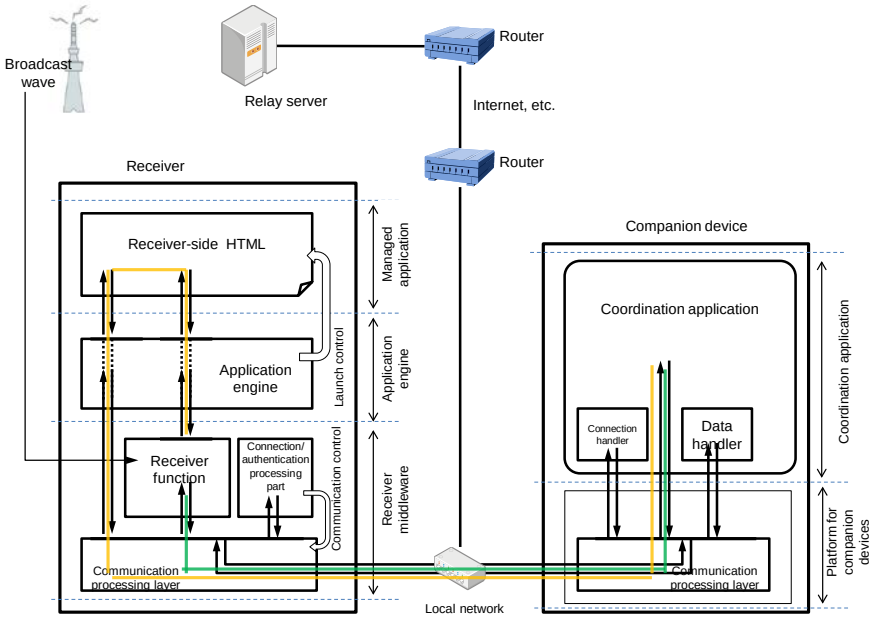


Fig. 12-25 Processing flow of System model A+B for collaboration with a companion device ⑤

12.1.3. Server relay method

In the server relay method, the receiver and the companion device concerned communicate with each other via a relay server in a network different from the ones to which they are connected.

A relay server may be installed, for example, by a service provider that provides the content for companion device collaboration. As shown in the reference model, it is usually assumed that the receiver and the companion device are connected to the same network as is the case with the direction communication method, but they can be connected to different networks.

12.1.3.1. System model C for collaboration with a companion device

In System model C, companion device collaboration is achieved through communication between a managed application that runs on the receiver and a collaboration application. The processing flow of System model C is shown below:

① Preparation of the collaboration application
The user of a companion device installs a collaboration application and launches it (Fig. 12-26).

② Reception of a broadcast video and launch of the managed application
The receiver receives a broadcast video and launched the managed application (Fig. 12-27).

③ Establishment of a connection between the managed application and a relay server
The managed application that has been launched establishes a connection to a relay server on its own. At that time, it authenticates the server and establishes a connection to it only when the authenticity of the server has been verified (Fig. 12-28).

④ Launch of the collaboration application and establishment of a connection between it and the receiver
The collaboration application installed in ① is launched. The connection handler is connected to the same relay server as was mentioned in ③. This relay server establishes a connection between the managed application and the collaboration application (Fig. 12-29).

⑤ Access to receiver functions by the collaboration application
In this model, the API used to access receiver functions is disclosed only to managed applications on the receiver. When the collaboration application wants to access receiver functions, it sends an access request using message communication between the collaboration application and the managed application. When the managed application receives the request, it operates the API, and sends the result to the collaboration

application if necessary (Fig. 12-30).

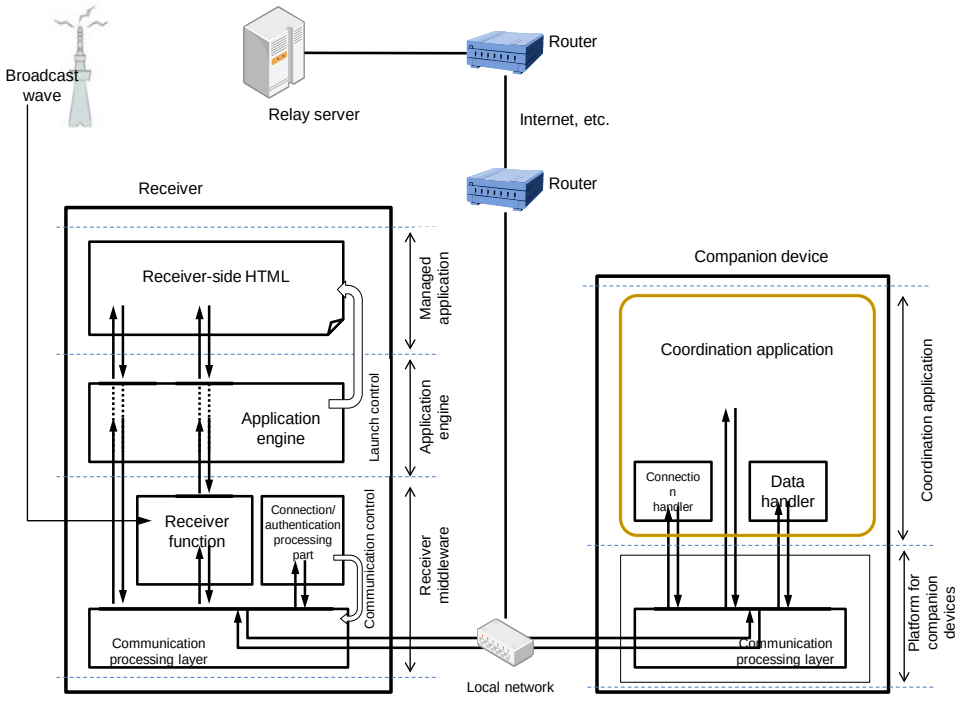


Fig. 12-26 Processing flow of System model C for collaboration with a companion device ①

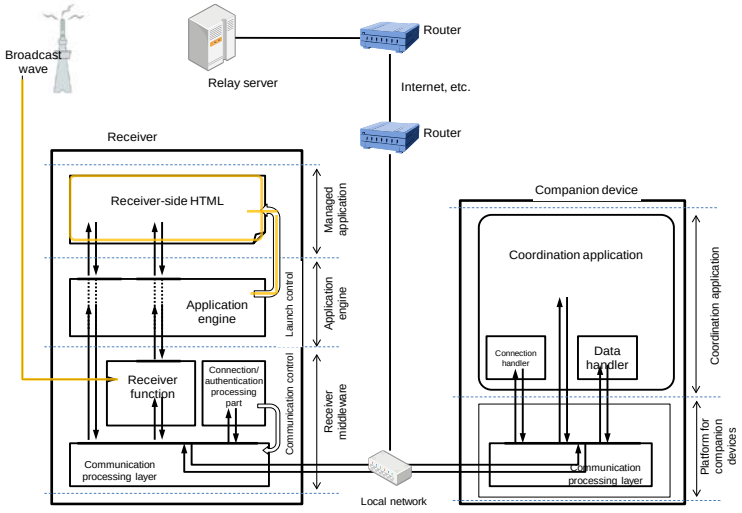


Fig. 12-27 Processing flow of System model C for collaboration with a companion device ②

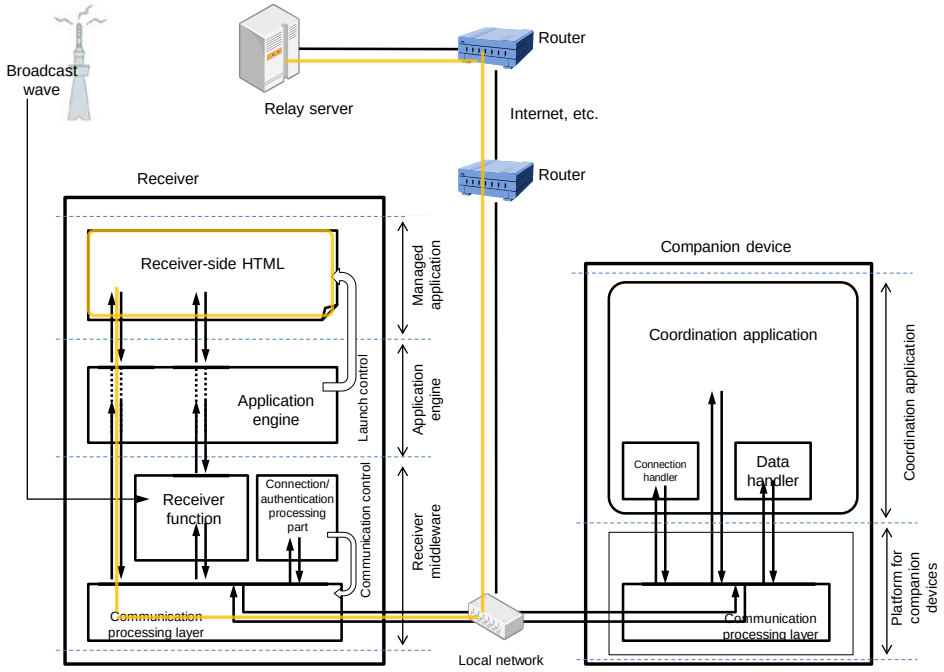


Fig. 12-28 Processing flow of System model C for collaboration with a companion device ③

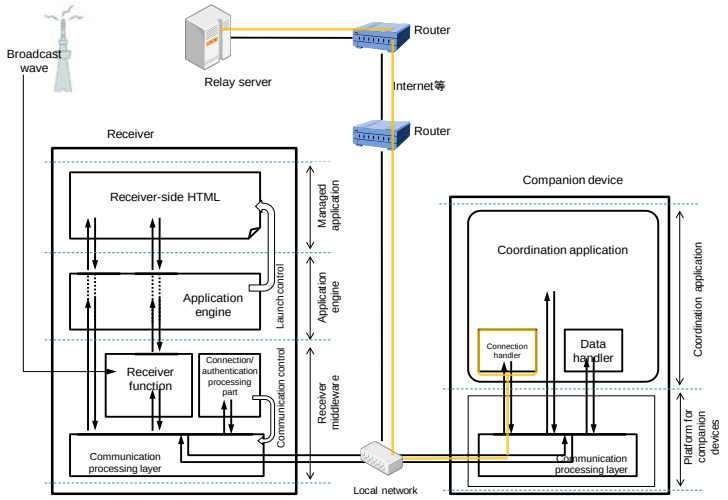


Fig. 12-29 Processing flow of System model C for collaboration with a companion device ④

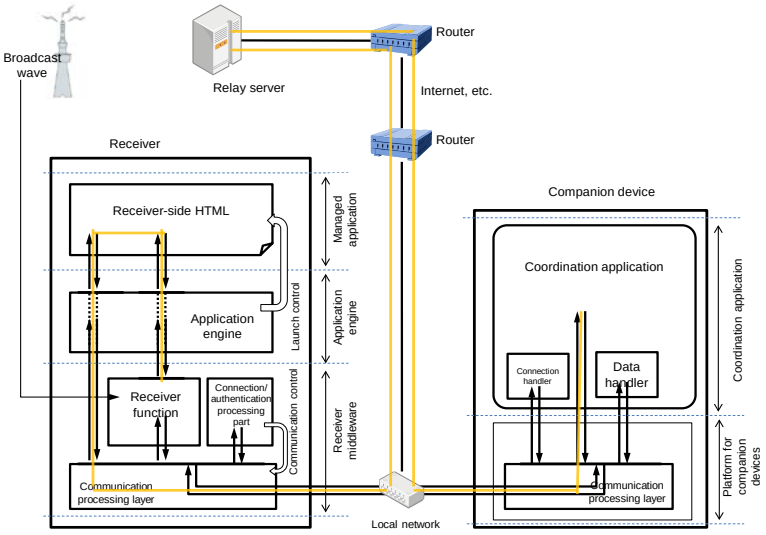


Fig. 12-30 Processing flow of System model C for collaboration with a companion device ⑤

12.1.3.2. System model D for collaboration with a companion device

In System model D, companion device collaboration is achieved through communication between a function built into the receiver and the collaboration application. The processing flow of System model D is shown below:

① Preparation of the collaboration application
The user of a companion device installs a collaboration application and launches it (Fig. 12-31).

② Establishment of a connection between the receiver and a relay server
The receiver establishes a connection to a relay server. At that time, it authenticates the server and establishes a connection to it only when the authenticity of the server has been verified (Fig. 12-32).

③ Launch of the collaboration application and establishment of a connection to the relay server
The viewer selects a collaboration application and launches it. The launched collaboration application is connected to the same relay server as in ② (Fig. 12-33). A connection between the receiver and the collaboration application is established via this relay server. When this connection is established for the first time, the receiver stores information about the companion device to which it is connected and the collaboration application as necessary (pairing), and registers the user with the relay server.

④ Access to receiver functions by the collaboration application
In this model, the API used to access receiver functions is disclosed to the collaboration

application as appropriate, and the collaboration application accesses receiver functions using this API via the relay server (Fig. 12-34).

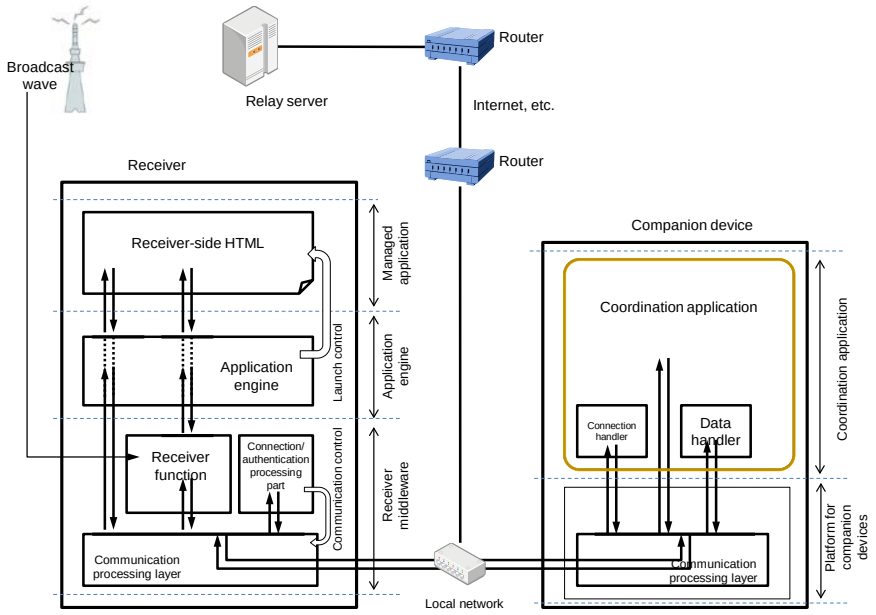


Fig. 12-31 Processing flow of System model D for collaboration with a companion device ①

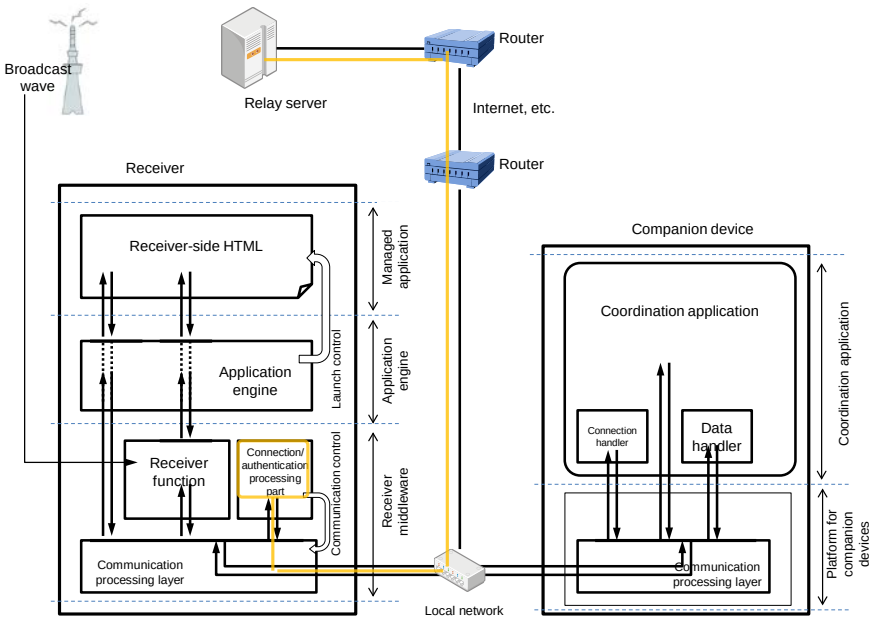


Fig. 12-32 Processing flow of System model D for collaboration with a companion device ②

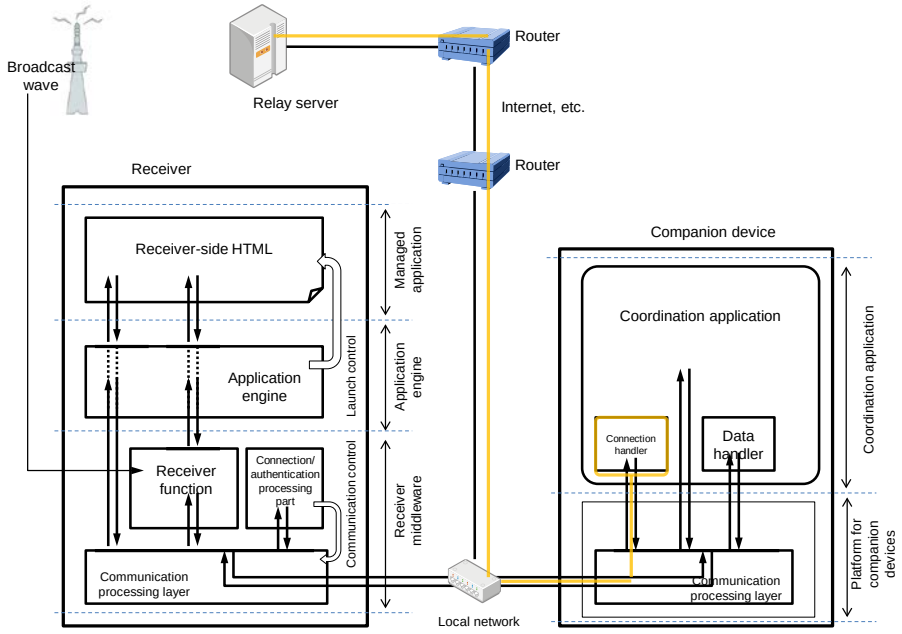


Fig. 12-33 Processing flow of System model D for collaboration with a companion device ③

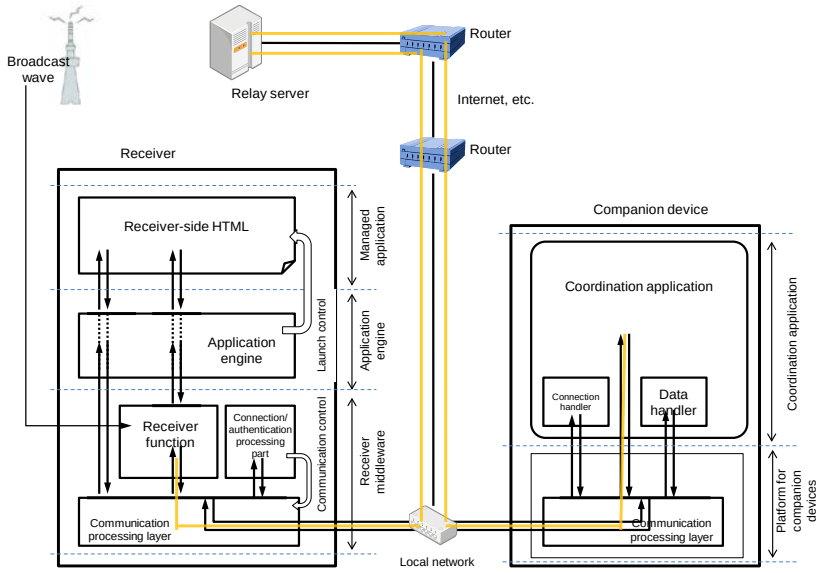


Fig. 12-34 Processing flow of System model D for collaboration with a companion device ④

12.2. Interface between the receiver and a companion device

The classification of system models for collaboration with a companion device described in the previous section are based on differences in how the receiver and the companion device are related to each other. This is the reason why the reference model described in 12.1.1 makes no distinction

regarding internal implementation of receiver functions, which consist of a variety of functions, and treats the collaboration application as a generic entity although its internal implementation can vary.

However, which protocol is appropriate for communication between the receiver and the companion device in each system model depends on which receiver function communicates at each processing stage, and how the collaboration application is internally implemented.

In light of the above, this section describes interface models that are used to specify the interface between the receiver and the companion device. Which specific protocol is to be applied to the interface of each model is to be specified in the future.

12.2.1. Direct communication method

12.2.1.1. Detailed model for receiver functions in the direct communication method

A detailed receiver model specifying the interface between the receiver and the companion device in the direct communication method is shown in Fig. 12-35. Each of the entities indicated by ① to ⑧ in the figure is an end point of the interface. It is necessary to specify each interface in a way that suits the internal implementation of the collaboration application that provides the other end points of the interface.

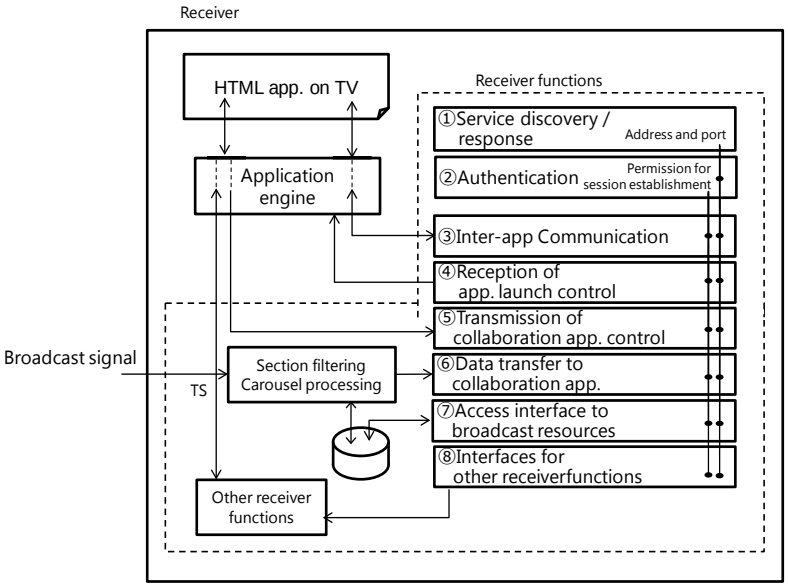


Fig. 12-35 Detailed model for receiver functions in the direct communication method

The interface between each entity and the collaboration application is described in the following sections. The three examples of implementation of the collaboration application described in 12.1.1 are considered.

12.2.1.2. Service discovery

In cases where the collaboration application is a native-type or a browser-type application, the communication method and data format that are to be used for communication between the service discovery/response entity and the collaboration application (connection handler) are specified for the interface. In cases where the collaboration application is a browser type, a communication protocol is selected for the interface from among those supported by the browser. The receiver's service discovery/response entity needs to implement this protocol. There is no restriction on data format. It can even be a proprietary format.

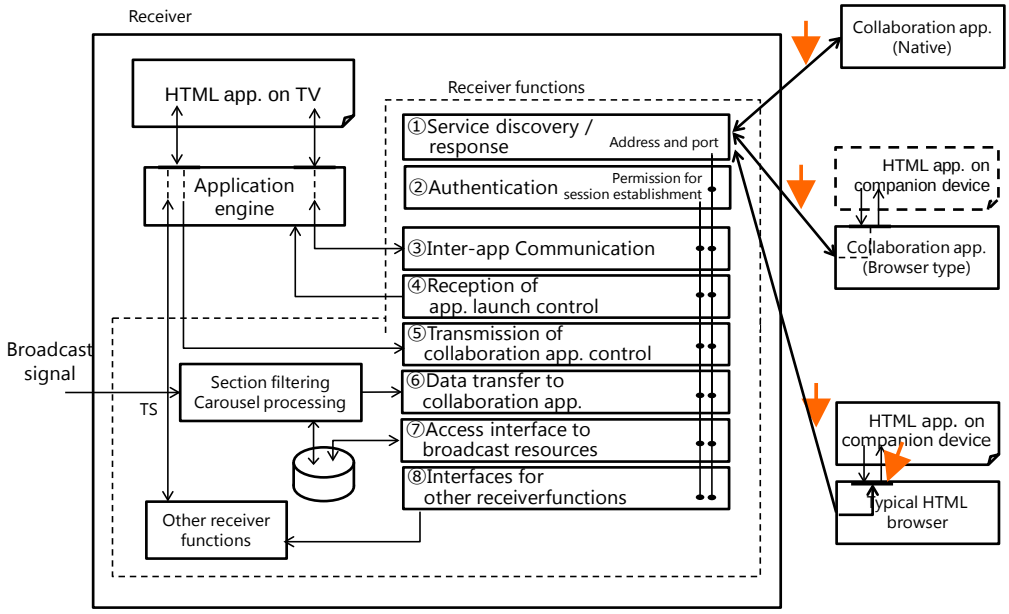


Fig. 12-36 Interface model for service discovery/response in the direct communication method

12.2.1.3. Authentication

In cases where the collaboration application is a native-type or a browser-type application, any appropriate communication method and data format, including a proprietary one, can be selected for communication between the authentication entity and the collaboration application (connection handler) and specified for the interface. However, if HTML content that runs on a browser-type application is also to be authenticated, it is necessary to specify the API provided by the application as well.

In cases where the collaboration application is a browser type, a communication protocol and a browser API are selected from among those supported by the browser and specified for the interface.

The receiver's authentication entity needs to implement this protocol. The data format may be determined as a part of the selected communication protocol in some cases. Any appropriate data format, including a proprietary one, may be selected in other cases.

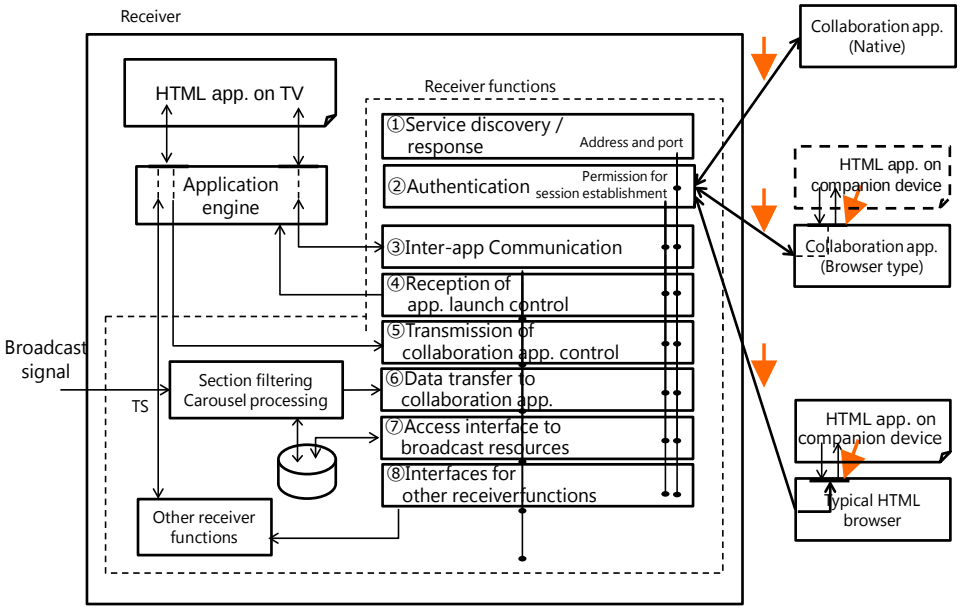


Fig. 12-37 Interface model for authentication in the direct communication method

12.2.1.4. Inter-application communication

In cases where the collaboration application is a native type, any appropriate communication method and data format, including a proprietary one, can be selected for communication between the inter-application communication entity and the collaboration application (connection handler) and specified for the interface. This is also true for cases where the collaboration application is a browser-type application. However, in these cases, it is also necessary to specify the API used by HTML for communication.

In cases where the collaboration application is a browser type, a communication protocol capable of exchanging any data and a browser API are selected from among those supported by the browser and are specified for the interface. The receiver's inter-application communication entity needs to implement this protocol. Any appropriate data format, including proprietary one, may be selected.

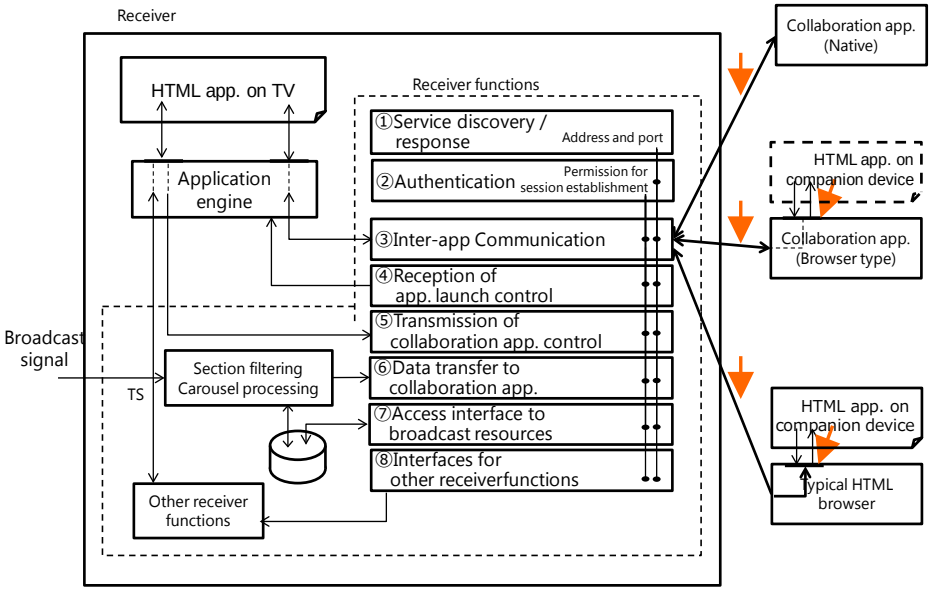


Fig. 12-38 Interface model for inter-application communication in the direct communication method

12.2.1.5. Reception of application launch control

This is the same as in inter-application communication described in 12.2.1.4.

12.2.1.6. Transmission of collaboration application launch control

This is the same as in inter-application communication described in 12.2.1.4.

12.2.1.7. Transfer of data for the coordinated application

This is the same as in inter-application communication described in 12.2.1.4.

12.2.1.8. Access to broadcast resources by the collaboration application

This is the same as in inter-application communication described in 12.2.1.4.

12.2.1.9. Access to other receiver functions

Remote control of the receiver and pointer operation functions, etc. using a companion device.

This is the same as in inter-application communication described in 12.2.1.4.

12.2.2. Server relay method

TBD

Chapter 13. Non-Broadcast-Oriented Managed Applications

This chapter specifies system models, the classification and lifecycles of applications, and application authentication models for non-broadcast-oriented managed applications.

A non-broadcast-oriented managed application is a managed application that is permitted to access broadcast resources based on an authentication mechanism without using the broadcast signal. Unlike the case with broadcast-oriented managed applications, which are specified in Chapter 7, the applications are usually managed by receiver functions through broadband communication, and the launch of an application is usually initiated by a user operation. Since the launch of a non-broadcast-oriented managed application is initiated by a mean other than the broadcast signal, it is necessary that an application is authenticated before it accesses to broadcast resources.

13.1. System Model

A functional system model for non-broadcast-oriented managed applications is shown in Fig. 13-1. The figure details the part of the system model described in Fig. 13-1and it is not intended to specify the specific operation and administration. The specific operation and administration are determined in a contract/agreement between the broadcaster, platform provider and application developers concerned.

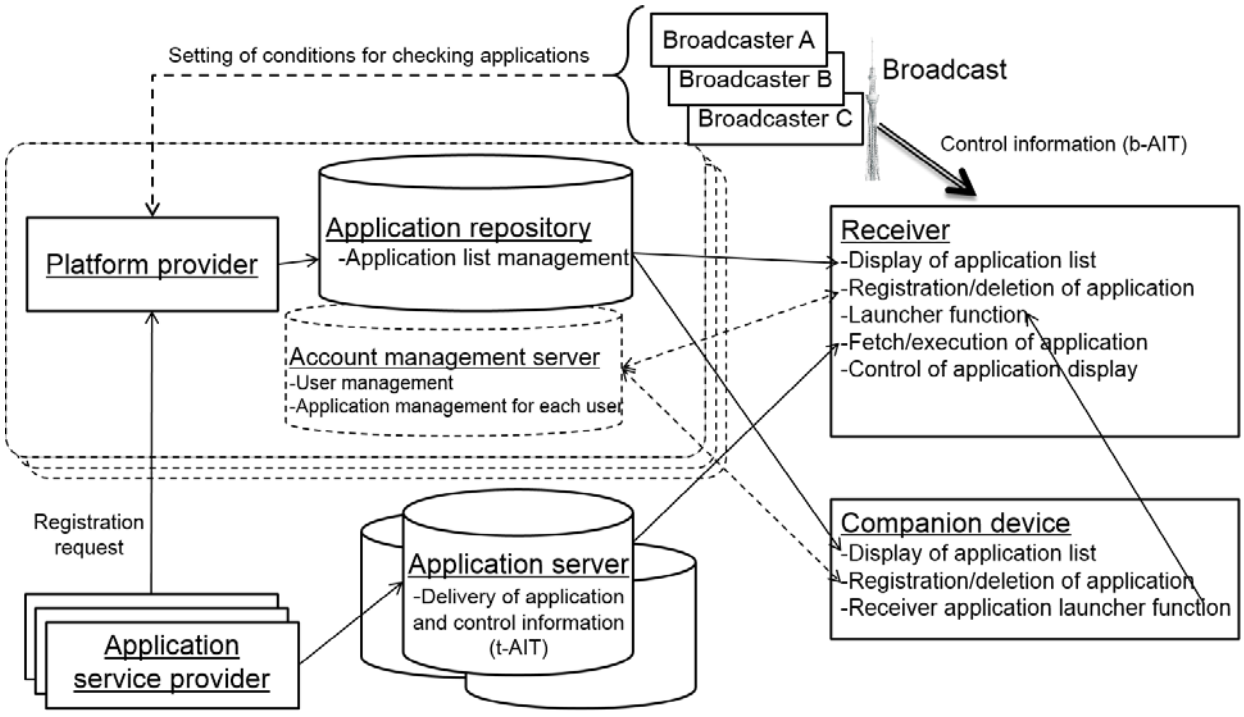


Fig. 13-1 System model for non-broadcast-oriented managed applications

This system model consists of the following four elements;

- Platform provider

  A platform provider can be one of the elements defined as a service provider in Chapter 5. It is a functional element that manages this system entirety. It operates an application repository that manages the non-broadcast-oriented managed applications to be delivered, and provides an application list to the receiver and companion devices. This application list contains application names, IDs, and URLs by which each application can be acquired, etc.. It is assumed that, before an application is registered in the list, the broadcaster checks whether the application satisfies conditions. A platform provider may manage user accounts and manage the use of applications by individual users.

- Application/service provider

  An application/service provider is another element defined as a service provider in Chapter 5. It is a functional element that delivers non-broadcast-oriented managed applications and provides services to applications that are runnning. Before delivering an application, the service provider needs to register it with the platform provider. The service provider delivers, along with the application itself, application control information (t-AIT, which is specified in Chapter 14) that has been created or confirmed by the platform provider. In some cases, the platform provider may deliver both applications and their associated t-AITs.

- Broadcaster

  In addition to provision of broadcast programs over a broadcast signal, a broadcaster also transmits the control information table (b-AIT: see Appendix F) to inform the receiver of policies on whether an application is allowed to run and whether it is allowed to access broadcast resources, and policies on presentation control. It also sets up conditions regarding acceptance of applications for platform providers.

- Receiver

  A receiver is a functional element that executes applications. In some cases, the user may launch or operate applications using a companion device connected to the receiver. The required functions for a receiverare described in 13.6.

13.2. Application class

Applications are classified according to the way in which the screen is controlled during their execution and according to the way they are delivered. The two types of class are specified in 13.2.1 and 13.2.2, respectively.

13.2.1. Layout classes

The following three layout classes are defined according to the way in which the screen is controlled while the application concerned is executed. Parallel execution of multiple applications are allowed

depending on the category the application concerned belongs to. Details about multiple applications is specified in 13.7.

1. Broadcast video inclusion class
   A broadcast video inclusion class application can control display of the entire screen. It can refer to the broadcast video and control the layout of the entire screen including size and position of the broadcast video. Access to broadcast resources, such as referring to the broadcast video or accessing to SI information, is permitted depending on the scope of permitted operations (hereafter called the “operation scope”) described in 13.3. An application may overlay on the broadcast video, or render graphics components of the application around the reduced sized broadcast video.
2. Widget class
   A widget class application occupies a part of the screen. Its position on the screen is determined by the receiver. Although the application is not allowed to refer to the broadcast video, access to broadcast resources, such as accessing to SI information, can be permitted in accordance with the operation scope setting described in 13.3. It may overlay on the broadcast video, or be allocated in a distinguished area on a screen not to disturb presentation of the broadcast video. The receiver controls the layout of the entire screen based on the application’s control information provided through broadband and/or broadcast.
3. Non-display class
   A non-display class application has no displayable components, and executes only a script. Access to broadcast resources, such as accessing to SI information, is permitted in accordance with the operation scope setting described in 13.3.

13.2.2. Package classes

The following two classes are defined according to the way in which an application is delivered. The operation for registering an application with an application launcher and the procedure for application authentication differ between the two.

1. Packaged class
   A packaged class application is installed in the receiver from an application delivery server at first, and then it is launched by the receiver’s launcher. Specific details of how the application is packaged and how it operates are to be specified in the future.
2. Host class
   It is not necessary to install a host class application. The selection of an application on receiver’s launcher launches the application and retrieves the top page of the application.

13.3. Life cycle

The life cycle of a non-broadcast-oriented managed application is shown in Fig. 13-2. The states, state transition events, and the processing shown in this figure are described below. Note that this section describes only the state transitions of a single application. Restrictions and inter-application interactions for concurrent execution of multiple applications are specified in 13.7.

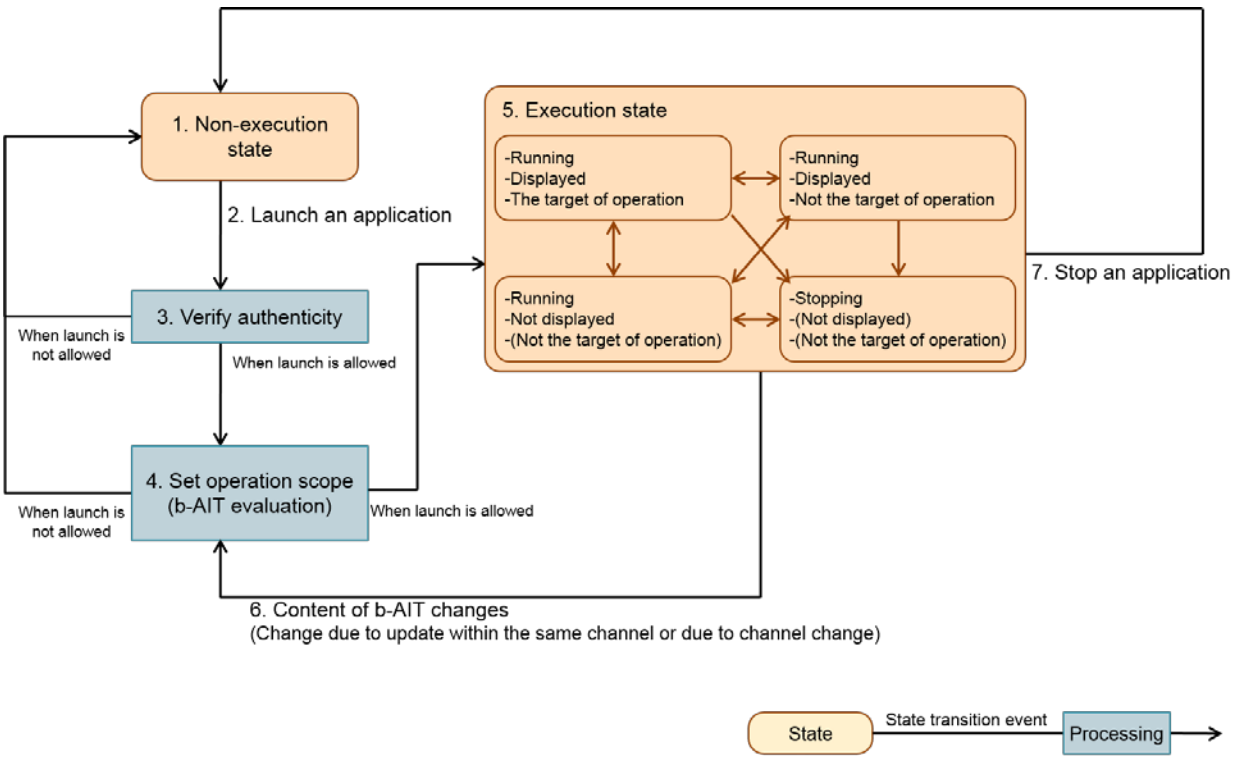


Fig. 13-2 Life cycle of a non-broadcast-oriented managed application

① Non-execution state

In the non-execution state, the application concerned is ready to be executed by the receiver but it is not yet loaded into the application engine and has not been launched. What is assumed by “the application being ready to be executed” is that a packaged class application has already been installed in the receiver or that a host class application has been registered in the receiver’s bookmarks.

② Launch an application

This is an operation or an event that makes an application in the non-execution state get into launch sequence as specified in Step ③ and subsequent steps. Details of launching is specified in 13.4.1.

③ Verify authenticity of the application

The authenticity of an application is verified using a method described in 13.5. If the verification is successful and it has been determined that the application can be launched, the processing moves to Step ④. Otherwise, the application is not launched, and the processing returns to State ①. For more details of this step, see 13.4.2.

④ Set the permitted operation scope of the application

Based on the latest b-AIT that is multiplexed in the broadcast signal being selected, it is determined whether it is allowed to launch the application and, if it is allowed, the application's operation scope is set. If this processing is to be re-applied to an application that is already being executed, it is determined whether it is allowed to continue the execution. If it is allowed, the application moves to State ⑤. For more details of this step, see 13.4.2.

⑤ Execution state

This is the state in which the application concerned has been launched and is being executed. An application in execution state takes one of the sub-states described below depending on its operation state (running or not), its display state (displayed or not), and whether it is the target of manipulation. The receiver determines which sub-state should be taken based on the operation scope set in Processing ④, the states of other applications that are simultaneously in the execution state, and the user's manipulations. For the case where multiple applications are in the execution state, see 13.7.

(1) Running, displayed, and the target of manipulation

This is a sub-state in which the application concerned is running, the application occupies display area on the screen, and the application is the target of the user's manipulation.

(2) Running, displayed, and not the target of manipulation

This is a sub-state in which the application is running and is the application occupies a display area on the screen but the application is not the target of the user's manipulation.

(3) Running and not displayed

This is a sub-state in which the application is running, the application occupies no display area on the screen, and the application is not the target of the user's manipulation.

(4) Not running

This is a sub-state in which the application is in execution state but is temporarily not running. The application occupies no display area on the screen, and the application is not the target of the user's manipulation.

⑥ Update of the b-AIT for an application being executed

When the content of the b-AIT multiplexed in the broadcast signal of the channel being selected is updated while the application is in an execution state, or when the broadcast channel selected is changed, Step ④ is executed anew based on the content of the updated b-AIT.

⑦ Stop an application

This is an operation or an event that puts an application in an execution state to the non-

execution state ①. The specific method of stopping an application is specified in 13.4.3.

13.4. Launching or stopping an application

13.4.1. Methods of launching an application

A non-broadcast-oriented managed application is launched in one of the following ways.

- Launch by the application launcher
  The user launches an application using an application launcher (see 13.6) that has been implemented as a receiver function.
- Launch on an instruction given by a companion device
  The application is launched under application launch control (see 12.2) of a companion device communicating with the receiver.
- Automatic launch based on conditions preset by the user
  When the user registers an application with an application launcher, he/she may set launch conditions. For example, when the receiver is turned on, the application is launched without the user's direct manipulation.
- Launch by a broadcast-oriented managed application
  An application is launched when a broadcast-oriented managed application calls replaceApplication().
- Launch from a non-broadcast-oriented managed application
  An application is launched when a non-broadcast-oriented managed application calls replaceApplication().
- Launch from a data broadcast browser
  An application is launched when a data broadcast browser executes the startAITControlledApp() .

13.4.2. Launch sequence

The sequence after an application has been instructed to launch through one of the methods described in 13.4.1 is shown in Fig. 13-3. This figure shows Steps ③ and ④ described in 13.3 in more detail.

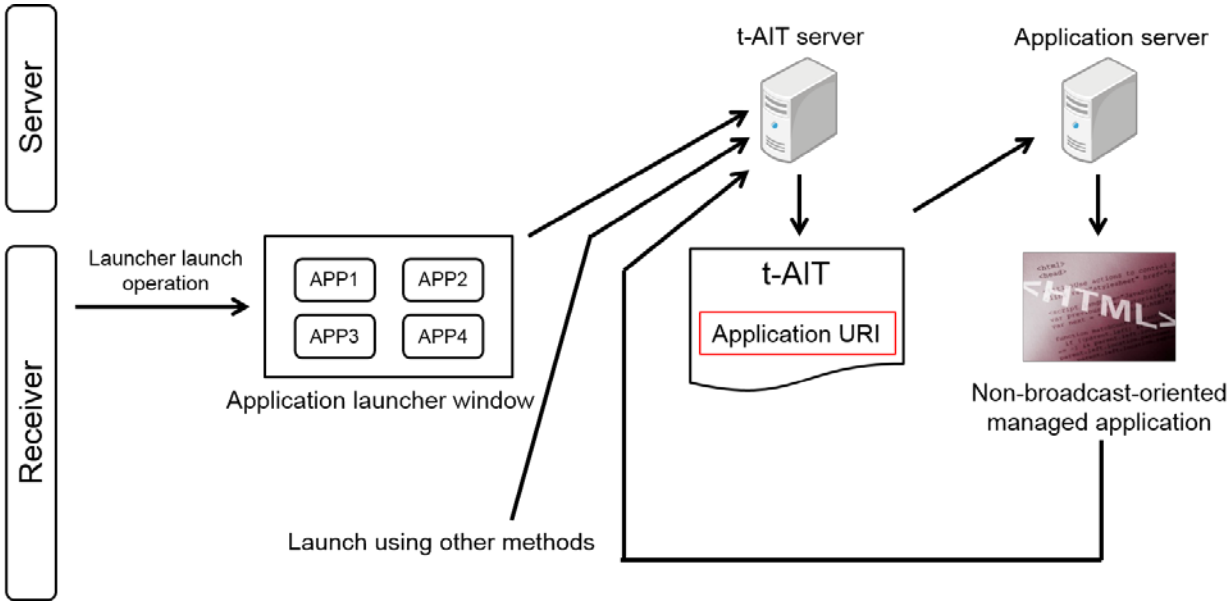


Fig. 13-3 Launching sequence of a non-broadcast-oriented managed application

1. When the application launch process is started using a method described in 13.4.1, the receiver retrieves the associated t-AIT. It then acquires the application from the URL specified in the t-AIT, and launches it. If signature verification is required for the t-AIT and the application, this is in accordance with one of the methods described in 13.5.
2. The receiver binds the application and the broadcast service, and controls the application based on the access control description in the t-AIT. For example, if the application attempts to access a broadcast resource it is not authorized to access, the receiver blocks this access. If access control for the non-broadcast-oriented managed application is written in its b-AIT, the receiver integrates access control of the two AITs. For example, the receiver blocks any operation that is permitted in the t-AIT but not permitted in the b-AIT.
3. When the b-AIT is updated or when another broadcast service is selected, the t-AIT and b-AIT are re-evaluated as necessary to update the access control conditions. If an instruction is issued to make access that is not permitted as the result of the re-evaluation of AITs, the access is stopped.

13.4.3. Methods of stopping an application

A non-broadcast-oriented managed application is stopped in the following ways or through the following events:

- Instruction to stop an application through a user interface provided by a receiver function
  When the user executes a stop operation implemented in an application launcher, the

application is stopped.

- Self termination by the application

  An application stops when it calls the destroyApplication().

- Transition to another application

  A non-broadcast-oriented managed application is stopped by a call of replaceApplication().

- Transition to a general application

  A non-broadcast-oriented managed application is stopped by a call of exitFromManagedState().

- Transition of an application to outside the boundary or execution of an operation that is not permitted

  When there is a transition to outside the boundary according to the settings in the t-AIT and b-AIT, or when an operation that is not permitted is executed, the application manager (see 13.6) may stop the execution of the application.

- Stopping of an application through other events

  The application manager may stop the execution of an application by some events, such as the number of running applications exceeds a limit. It is desirable that the limit of the number of running applications and other restrictions regarding the execution of applications are specified in operational guidelines.

13.5. Authentication of an application

Since non-broadcast-oriented managed applications are launched or stopped outside the control of the broadcast signal, it is necessary to authenticate applications to determine whether it is permitted to launch them or whether they are permitted access to broadcast resources.

13.5.1. Methods of authenticating an application

The following three authentication methods are defined as ways to guarantee the source or authenticity of non-broadcast-oriented managed applications. An individual application may be authenticated using a single method or a combination of several methods. However, note that when there is a mix of applications that are authenticated using only Method 1 and applications that are authenticated using another method or using a combination of methods, there may be a security risk, such as alteration of signature and attempts to lead the applications to improper servers.

Method 1. Method using SSL/TLS

This method prevents alteration of an application list retrieved from a repository and applications retrieved from an application server at somewhere in the broadband paths. In addition, it prevents spoofing of application servers. Mechanisms for verifying that the servers concerned, such as a repository, are authenticated shall be specified in operational guidelines. (see Appendix H (1))

Method 2. Method using a digital signature (1)

This method uses a digital signature to prevent alteration of t-AITs. It guarantees the authenticity of an application based on the reliability of its t-AIT. (see Appendix H (2))

Method 3. Method using a digital signature (2)

This method uses a digital signature to prevent alteration both of t-AITs and of applications. It also guarantees the source of an application using a certificate issued by a platform provider. (see Appendix H (3))

13.5.2. Basic model for application authentication

A basic model for application authentication is shown in Fig. 13-4. This figure covers all the three authentication methods described in the previous section. This model details the application authentication part of the system model for non-broadcast-oriented managed applications described in 13.1. Note that the assumptions made in this model as to providers and the relationship between elements and actual functions are not intended to specify how they are actually implemented. These shall be determined to suit the actual operation.

This model consists of the following elements and functions:

- Root certificate authority (CA)

  A pubic certificate issuing agency is assumed for the root certificate authority. The root certificate authority issues root certificates, server certificates, Certificate 1 and Signature Key 1 for platform providers and broadcasters.

- Platform provider

  This is a provider that provides integrated broadcast-broadband services. A platform provider may be a broadcaster, a conglomerate of multiple broadcasters, or a provider that provides a delivery service under the control of a broadcaster. It accepts requests for application registration from application developers, confirms behavior of applications and generates t-AITs. In addition, it holds Certificate 1 and Signature Key 1 issued by the root certificate authority, and attaches a signature to a t-AIT according to the intention of the broadcaster using Signature Key 1 in accordance with the specification given in Chapter 14. A t-AIT to which this signature attached is placed in an application server.

  In the case of Method 3, a platform provider issues Certificate 2 and Signature Key 2 for an application developer.

- Application developer

  An application developer registers applications they have developed to application repository. The developer needs to conclude a contract of accessing to broadcast resources and program-related information with platform providers in advance.. It is assumed that applications are placed in application servers by application developers themselves.

In case of Method 3, an application developer attaches a signature to an application using Signature Key 2 issued by a platform provider, and places the application in an application server along with Certificate 2. It is assumed that either a platform provider or a broadcaster attaches a signature to an application using Signature Key 2. Details of attaching a signature to an application shall be specified in operational guidelines.

- Broadcaster

  A broadcaster provides a broadcast service. It provides programs over a broadcast wave, multiplexes Certificate 1 on the broadcast signal, and transmits the broadcast signal to receivers. It also informs a platform provider of its intention regarding the use of broadcast resources and program-related information, etc.

- Application repository

  This is a central server that manages an application list, and provides the user with a list of available applications and information about where they can be retrieved from. Application repositories are administered by platform providers.

  In the case of Method 1, an application repository has a server certificate issued by the root certificate authority.

- Application server

  This is a server in which applications themselves and their t-AITs are stored. The server(s) delivers them in response to HTTP/HTTPS requests from receivers. A variety of ways of operation (who is the managing entity and how the operation is carried out) can be implemented.

  In the case of Method 1, an application server holds a server certificate issued by the root certificate authority.

- Receiver

  A receiver receives the broadcast signal, retrieves an application list and t-AITs, and retrieves and executes applications. A root certificate has to be pre-installed. It receives Certificate 1, which is multiplexed on the broadcast signal, as necessary, verifies Certificate 1 using the root certificate prior to other operations, and stores it in non-volatile memory. Before a receiver executes an application, it verifies the signature attached to its t-AIT using a verification key contained in Certificate 1. If the verification is successful, it retrieves the application from the URL of the application's entry document written in its t-AIT, and launches the application. If the verification fails, the receiver displays a message on the screen, and does not retrieve the application. If an application hash value is contained in a t-AIT, it is verified simultaneously.

  In the case of Method 3、after retrieval of an application as above, the receiver goes on to verify Certificate 2 attached to the application using a verification key contained in Certificate 1. If the verification is successful, the receiver verifies the signature attached to the application using a verification key contained in Certificate 2. If this verification is also successful, it launches the

application. If either of these verifications fails, the receiver behaves as described above for the case of verification failure.

In the case of Method 1, when the receiver retrieves an application list and t-AITs, etc., it shall authenticate the server concerned using a root certificate.

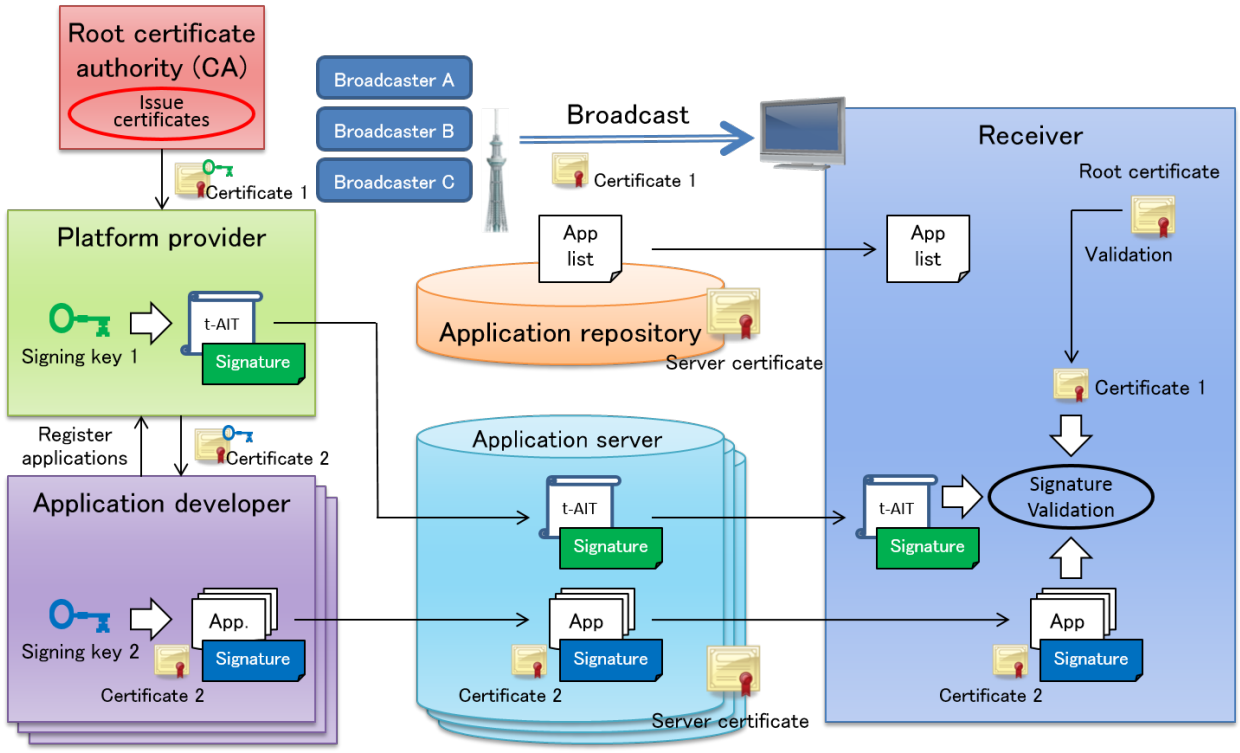


Fig. 13-4 Basic model for application authentication

13.6. Receiver model

Figure 13-5 shows a functional model related to non-broadcast-oriented managed applications as the part of the receiver model specified in 6.2.The figure shows how each functional block interacts with external entities. Note that this figure is not intended to specify a specific receiver implementation.

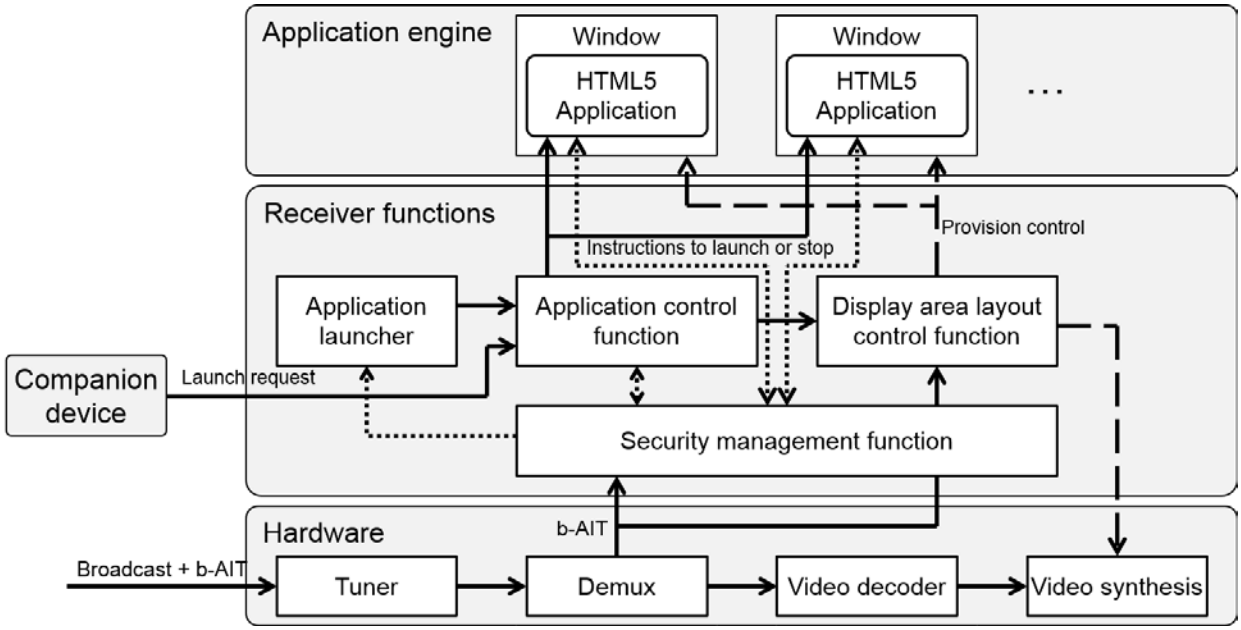


Fig. 13-5 Functional receiver model related to non-broadcast-oriented managed applications

- Application launcher

  This is an entity that provides a means by which the user can launch a non-broadcast-oriented managed application.

  An application launcher comprises a user interface for listing available applications provided by an application repository, selection of an application to register or delete it, and selection of an application from among the registered applications to launch it. When the user requests the launch of an application, an application launcher notifies the application control function of the URL in the application's t-AIT.

  Some application launchers may display only those applications that the broadcast service being received permits execution based on a determination made by the security management control function.

- Application control function

  This is an entity that controls the launching and stopping of non-broadcast-oriented managed applications. When a request for launching an application is issued in accordance with a method specified in 13.4.1, the application control function retrieves the application's t-AIT from the URL of the t-AIT, and issues to the application engine a command to launch the application from the URL specified by the t-AIT. This is executed only when the security management control function permits the launch of this application. When a request to stop an application is issued in accordance with a method specified in 13.4.3, the application control function similarly issues a stop command to the application engine.

  In addition, the application control function allots/releases a window of the application when an

application is to be launched/stopped, and notifies the display area layout control function of the result of this allotment/release.

- Display area layout control function

  This is an entity that controls how the broadcast video and a non-broadcast-oriented managed application are presented on a screen. When an application is to be launched, this function controls the broadcast video plane and the window of the application engine to meet the conditions given in the t-AIT and b-AIT of the broadcast service being received. The function takes a similar action when the presentation policy of the b-AIT is changed.

- Security management control function

  This is an entity that guarantees the source and security of non-broadcast-oriented managed applications, and restricts the functionality of applications as necessary, based on specified security management rules.

  This function authenticates the server certificates of application repositories and application servers specified in13.5 and signatures attached to t-AITs and applications. This function notifies the application control function of the result.

  When an application is to be launched, it determines whether the launch of this application is permitted based on the b-AIT that has been received, and notifies the application launcher and the application control function of the result. The function takes a similar action when the b-AIT is changed.

  When a running application attempts to access broadcast resources, the security management control function determines whether this access is permitted based on the operation scope described in 13.3. If the access is not permitted, the function blocks the access.

- Application engine (HTML browser)

  This is an entity that executes applications. It shall be capable of concurrent execution of multiple applications. An operation model for concurrent execution of multiple applications is specified in 13.7.

13.7. Model for concurrent execution of multiple applications

This Specification assumes that multiple non-broadcast-oriented managed applications will be launched and will operate in parallel, and further that these applications will also operate in parallel with data broadcast browsers and broadcast-oriented managed applications. In light of this assumption, this section specifies an operation model that allows multiple applications to run in parallel. In addition, to the extent necessary, this section specifies the required behavior of broadcast-oriented managed applications and data broadcast browsers.

13.7.1. Relation with broadcast-oriented managed applications and data broadcast browsers

It is supposed that non-broadcast-oriented managed applications are produced mainly by entities

other than broadcasters. These applications can access broadcast resources under permission of the broadcasters to which they are bounded. Therefore, any operation model that allows operations of non-broadcast-oriented managed applications to block the broadcasters' operations or presentations of broadcast-oriented managed applications or data broadcast browsers is not desirable. For this reason, a principle of the specification in this section is that non-broadcast-oriented managed applications operate without stopping the operations of broadcast-oriented managed applications or a data broadcast browser.

13.7.2. Possible combinations of applications running in parallel

The combinations of non-broadcast-oriented managed applications, broadcast-oriented managed applications and data broadcast browsers are assumed to operate in parallel. These combinations are shown in Fig. 13-6.

Integrated broadcast-broadband services provided in accordance with this Specification need not support all these combinations. They can select or restrict combinations that they will support.

- No more than one data broadcast browser can be in the launch state. In other words, it is assumed that multiple data broadcast browsers cannot run simultaneously.
- No more than one broadcast-oriented managed application can be in the launch state. In other words, it is assumed that multiple broadcast-oriented managed applications cannot run simultaneously.
- It is assumed that a data broadcast browser and a broadcast-oriented managed application cannot run simultaneously.
- No more than one broadcast video inclusion-type application can be in the launch state. In other words, it is assumed that multiple broadcast video inclusion-type applications cannot run simultaneously.
- It is assumed that multiple widget-type applications may run simultaneously. The maximum number of widget-type applications that can run in parallel shall be specified in operational guidelines.
- It is assumed that one or more widget-type applications may run in parallel with a data broadcast browser or a broadcast-oriented managed application. The maximum number of widget-type applications that can run in parallel shall be specified in operational guidelines.
- It is assumed that multiple non-display-type applications may run simultaneously. The maximum number of non-display-type applications that can run in parallel shall be specified in operational guidelines.
- It is assumed that one or more non-display-type applications may run in parallel with a data broadcast browser or a broadcast-oriented managed application. The maximum number of

non-display-type applications that can run in parallel shall be specified in operational guidelines.

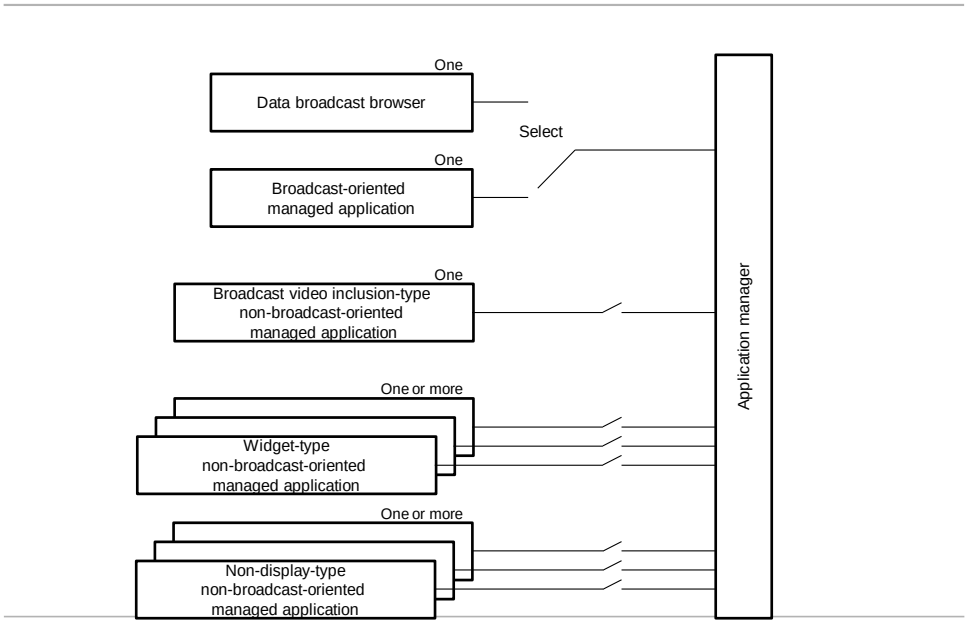


Fig. 13-6 Combinations of applications that are assumed to be able to run simultaneously

13.7.3. Method of displaying multiple applications

This section describes screen layouts in cases where one of the combinations of applications described in 13.7.2 are to run in parallel. Note that this section does not refer to non-display-type applications because they do not affect the screen layout.

13.7.3.1. Layout control

When multiple applications are to run in parallel, the receiver shall determine their layouts on the screen in such a way that the layouts will meet the conditions (hereafter referred to as “layout conditions”) determined by each application’s layout class and display size written in its t-AIT, and the video overlay control area written in the b-AIT. See Chapter 14 for t-AIT, and Appendix F for b-AIT.

When the application engine is to allot a display area for any data broadcast browser, broadcast-oriented managed application or broadcast video inclusion-type application, the application engine shall behave as if it allots an application a display area of the position and size specified by the application irrespective of the actual position and size of the area (on the display device) allotted by the receiver. Even when the receiver has changed the display area of an application while the latter is running, the above-mentioned application engine’s action makes it possible for the application to

continue to operate without any need to pay attention to the change. (However, this does not mean that the application should be prohibited from recognizing the change in the allotted display area and responding accordingly.) The video overlay control area written in the b-AIT shall be interpreted in such a way that the area allotted by the receiver to an application is the entire screen area, irrespective of the position and size of the display area of the broadcast video allotted by the applications.

13.7.3.2. Parallel execution of a broadcast-oriented managed application and widget-type applications

When a broadcast-oriented managed application and widget-type applications are to run in parallel, the receiver determines the display areas for the broadcast-oriented managed application and each widget-type application respectively and manages these areas. If the display areas of widget-type applications are to be overlaid on the display area of the broadcast-oriented managed application, this must be done in a way that satisfies the required layout conditions.

The display area of the broadcast video is within the display area of the broadcast-oriented managed application, and is determined by the broadcast-oriented managed application.

Screen layout examples for the above case are shown in Fig. 13-7 and Fig. 13-8. In both figures, two widget-type applications are simultaneously displayed together with a broadcast-oriented managed application. The layout conditions of the two figures are as follows:

① In Fig. 13-7, Application 1 is given permission in the t-AIT for overlay in the area allotted to the broadcast-oriented managed application. Since its size is small enough to fit in the area where overlay is permitted in the video overlay control information in the b-AIT, it is placed at the position shown in the figure. Application 2 is placed at the position shown in the figure because the t-AIT does not permit its overlay in the display area of the broadcast-oriented managed application or because it cannot fit in the video overlay control area set in the b-AIT.

② In Fig. 13-8, since there is no area where overlay is permitted in the setting in the b-AIT, both Applications 1 and 2 are allotted areas that do not overlap with the display area of the broadcast-oriented managed application.

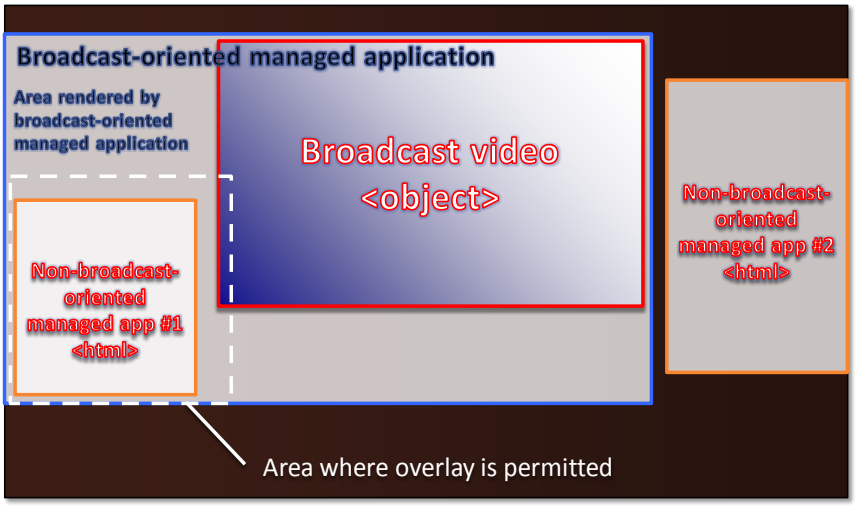


Fig. 13-7 Screen layout of a broadcast-oriented managed application and two widget-type applications ①

Fig. 13-8 Screen layout of a broadcast-oriented managed application and two widget-type applications ②

13.7.3.3. Parallel execution of a data broadcast browser and widget-type applications

In this case, the specification described in 13.7.3.2 applies with the words “broadcast-oriented managed application” replaced by “data broadcast browser.”

13.7.3.4. Parallel execution of a broadcast video inclusion-type application and widget-type applications

In this case, the specification described in 13.7.3.2 applies with the words “broadcast-oriented managed application” replaced by “broadcast video inclusion-type application.”

13.7.3.5. Parallel execution of a broadcast-oriented managed application and a broadcast video inclusion-type application

The following modes are assumed for cases where a broadcast-oriented managed application and a broadcast video inclusion-type application run in parallel. It is not necessary to support all these modes. Which modes may be used shall be specified in operational guidelines. If multiple modes are to be supported, it is desirable that the mode to use can be selected by the user. (For example, the user may switch between 3 and 4.).

1. Perfect simultaneous display
   This mode is equivalent to a case where both the broadcast-oriented managed application and the broadcast video inclusion-type application concerned are allotted display areas and operate independently of each other.
   The display area of the broadcast-oriented managed application shall not overlap with that of the broadcast video inclusion-type application.
2. Restricted simultaneous display
   In this mode, both the broadcast-oriented managed application and the broadcast video inclusion-type application concerned have their own display areas, but the operation of the broadcast-oriented managed application is restricted to some extent. One of the assumed restrictions is not displaying the broadcast video. Details of the restrictions shall be specified in operational guidelines.
   The display area of the broadcast-oriented managed application shall not overlap with that of the broadcast video inclusion-type application.
3. Broadcast video inclusion-type application not displayed
   In this mode, only the broadcast-oriented managed application is allotted a display area, and the broadcast video inclusion-type managed application is not displayed (i.e., it runs in background). The scope of operation that the latter application is permitted shall be specified in operational guidelines.
4. Broadcast-oriented managed application not displayed
   In this mode, only the broadcast video inclusion-type application is allotted a display area,

and the broadcast-oriented managed application is not displayed (i.e., it runs in background). The scope of operation that the latter application is permitted shall be specified in operational guidelines.

13.7.3.6. Parallel execution of a data broadcast browser and a broadcast video inclusion-type application

In this case, the specification described in 13.7.3.5 applies with the words “broadcast-oriented managed application” replaced by “data broadcast browser.”

13.7.3.7. Parallel execution of a broadcast-oriented managed application, a broadcast video inclusion-type application and widget-type applications

In this case, the modes described in 13.7.3.5 are applied to the broadcast-oriented managed application and the broadcast video inclusion-type application.

In Mode 1 or Mode 2, the receiver determines the display areas of the broadcast-oriented managed application, the broadcast video inclusion-type application and each widget-type application, and manages them. If a widget-type application is to be overlaid on the display area of the broadcast-oriented managed application, this should be done in a way that satisfies the required layout conditions. The display area of any widget-type application shall not overlap with that of the broadcast video inclusion-type application.

The specification in 13.7.3.2 is applied to Mode 3, and the specification in 13.7.3.4 to Mode 4.

13.7.3.8. Parallel execution of multiple widget-type applications

The receiver determines the display area allotted to the broadcast video and that allotted to each widget-type application, and manages them. A display area shall be allotted to each widget-type application in a way that satisfies the required layout conditions.

Screen layouts for this case are shown in Fig. 13-9 to Fig. 13-11. In all these screen layouts. In any of the figures, two widget-type applications are presented. These figures represent the following cases:

① In Fig. 13-9, both Applications 1 and 2 are permitted overlay in the t-AIT. The entire display area of the broadcast video is set in the b-AIT as an area where overlay is permitted. In this figure, the broadcast video is displayed at the maximum size, and the two applications are overlaid on the broadcast video.

② In Fig. 13-10, both Applications 1 and 2 are permitted overlay in the t-AIT. However, Application 1 is allotted an area that does not overlap with the display area of the broadcast video because Application 1 cannot fit in an area where overlay is permitted in the video overlay control information in the b-AIT even if the display area of the broadcast video is maximized and that of Application 1 is

minimized.

③ In Fig. 13-11, since no area is permitted overlay in the settings in the b-AIT, both Applications 1 and 2 are allotted display areas that do not overlap with the display area of the broadcast video.

Fig. 13-9 Screen layout for displaying multiple widget-type applications ①

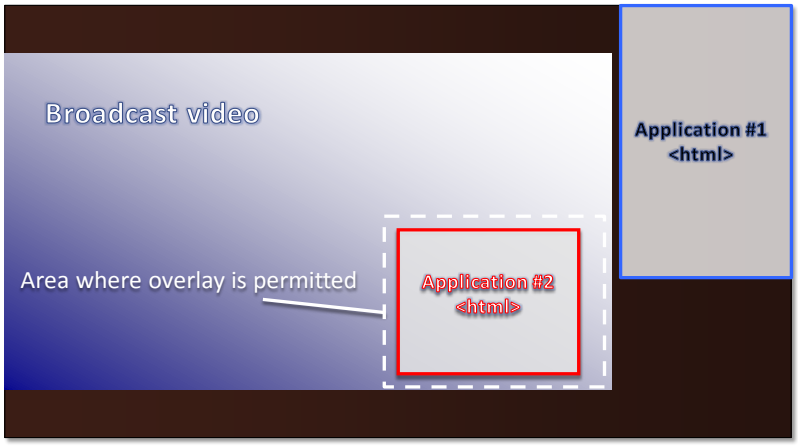


Fig. 13-10 Screen layout for displaying multiple widget-type applications ②

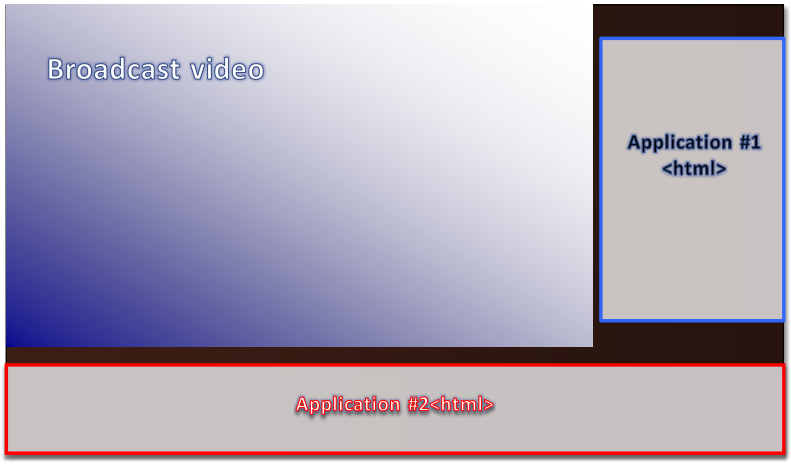


Fig. 13-11 Screen layout for displaying multiple widget-type applications ③

13.7.4. Selection of an application that is the target of the user's mainuplation

The receiver shall have a function that enables the user to select which application to manipulate from among simultaneously displayed applications (or a data broadcast browser). The method for selection of an application shall depend on the implementation of the receiver.

13.7.5. Arbitration of access by multiple running applications to receiver resources

This item is to be specified in the future.

Chapter 14. Application Control Information for Non-broadcast-oriented Managed Applications

This chapter defines XML-format AITs that are used to control non-broadcast-oriented managed applications. When a non-broadcast-oriented managed application is launched, the receiver must retrieve the XML-format AIT for it. If an XML signature is assigned to the application, the receiver shall retrieve the application from the location specified in the AIT only when the signature has been verified successfully. While a non-broadcast-oriented managed application is running, the application shall be controlled in accordance with the relevant XML-format AIT specified in this chapter. If non-broadcast-oriented managed application control information described in Appendix F is multiplexed in the broadcast signal, such information shall be used by application control together with the application control information specified in this chapter.

XML-format AITs conform to ARIB STD-B24 Volume 4 “Chapter 6 XML-format application control information” with some extension.

The structure of elements/attributes in the high-level structure of an XML-format AIT is shown in Table 14-1 and Fig. 14-1. The corresponding XML schema is shown in Table 14-2.

In cases where it is desired to distinguish the XML-format AIT specified in this chapter from the AIT that is multiplexed in the broadcast signal and described in Appendix F, the XML-format AIT specified in this chapter is called “t-AIT” in this Specification.

Table 14-1 High-layer structure of an XML-format AIT

Element/attribute name	Frequency	Definition
ServiceDiscovery	-	
@expirationDate	1	AIT expiration date
ApplicationDiscovery	1..n	Application discovery
@DomainName	1	Domain name
@Version	0..1	Version
ApplicationList	1..n	Application list
Application	1..n	Application

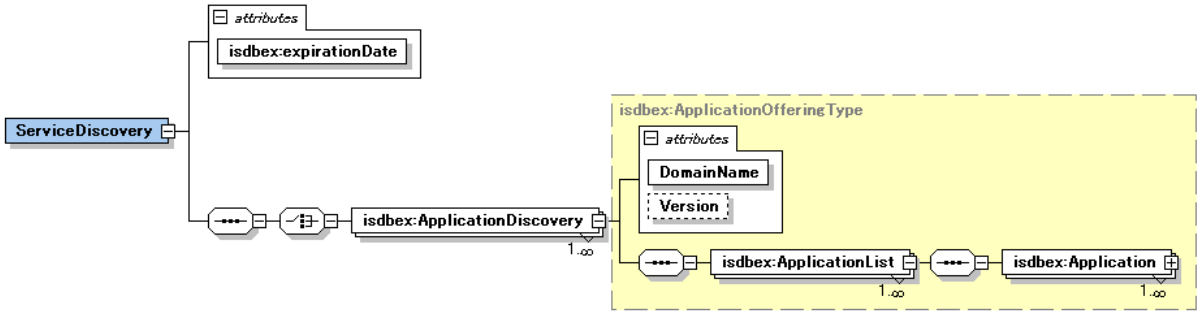


Fig. 14-1 High-level structure of an XML-format AIT

Table 14-2 XML schema of the high-level structure of an XML-format AIT

```
<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="isdbex:Application"
                                  maxOccurs="unbounded"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="ServiceDiscovery">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:choice>
        <xsd:element name="ApplicationDiscovery"
                        type="isdbex:ApplicationOfferingType" maxOccurs="unbounded"/>
      </xsd:choice>
    </xsd:sequence>
    <xsd:attribute name="expirationDate" type="xsd:date" use="required"/>
```

```
  </xsd:complexType>
</xsd:element>
```

In addition to ARIB STD-B24, the expirationDate attribute is added to the highest-level element, the ServiceDisovery element. This attribute shows the expiration date of the XML-format AIT. In the case of a non-broadcast-oriented managed application, what is written in the XML-format AIT needs to be authenticated. In the case of a package-type application, the terminal is assumed to hold the XML-format AIT as well. Therefore, it is necessary to specify the expiration date using this attribute. The terminal checks the expiration date. If the date has passed, the terminal considers this XML-format AIT to be invalid. If this invalid XML-format AIT is the one held by the terminal, the terminal refetches an XML-format AIT. The ApplicationDiscovery element exists below the ServiceDisovery element, and the ApplicationList element exists below the ApplicationDiscovery element. The elements mentioned so far conform in principle to ARIB STD-B24 Volume 4 Chapter 6. However, the Application element that exists blow the ApplicationList element, and elements below it have additional specifications as described below. The name space of isdbex is applied in principle to XML-format AITs specified in this Specification. However, as an exception, for elements that are confined to the specification of the source referred to, the name space of the source referred to is applied.

14.1. Application element

The Application element conforms in principle to ARIB STD-B24 Volume 4 “6.1 Application element” but the following specification is added. The structure of the Application element is shown in Table 14-3 and Fig. 14-2. The XML schema of the Application element is shown in Table 14-4.

Table 14-3 Structure of the Application element

Element/attribute name	Frequency	Definition
Application	-	
appName	0..n	Application name
applicationIdentifier	1	Application identifier
applicationDescriptor	1	Application descriptor
applicationLocation	1	Application location
applicationBoundary	0..1	Application boundary
applicationClass	0..1	Application class
applicationHash	0..1	Application hash
broadcastPermission	1..n	Broadcast permission

	appSize	0..n	Application display size

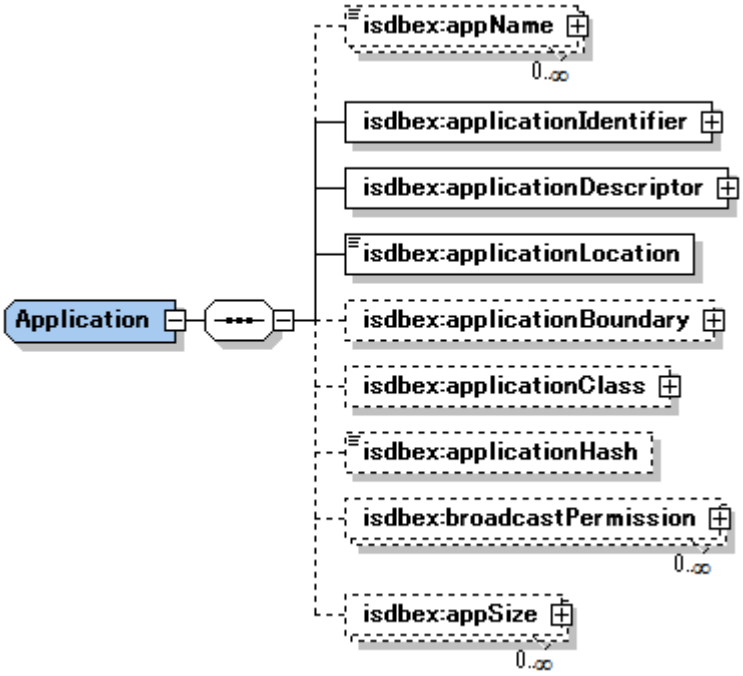


Fig. 14-2 Structure of the Application element

Table 14-4 XML schema of the Application element

```
<xsd:complexType name="Application">
  <xsd:sequence>
    <xsd:element name="appName" type="ipi:MultilingualType" minOccurs="0"
          maxOccurs="unbounded"/>
    <xsd:element name="applicationIdentifier" type="isdb:ApplicationIdentifier"/>
    <xsd:element name="applicationDescriptor" type="isdb:ApplicationDescriptor"/>
    <xsd:element name="applicationLocation"
          type="mhp:SimpleApplicationLocationDescriptorType"/>
    <xsd:element name="applicationBoundary"
          type="mhp:SimpleApplicationBoundaryDescriptorType" minOccurs="0"/>
    <xsd:element name="applicationClass" type="isdbex:ApplicationClassType"
          minOccurs="0"/>
    <xsd:element name="applicationHash" type="xsd:base64Binary" minOccurs="0"/>
    <xsd:element name="broadcastPermission" type="isdbex:BroadcastPermissionType"
          minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="appSize" type="isdbex:AppSizeType" minOccurs="0"
          maxOccurs="unbounded"/>
```

```
  </xsd:sequence>
</xsd:complexType>
```

Among the elements that conform to ARIB STD-B24 Volume 4 Chapter 6, the applicationIdentifier element and the applicationDescriptor element are applied under the Application element. The rest of the elements are not applied. However, the appName element, the applicationLocation element and the applicationBoundary element that 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,” Section “5.4 XML-based syntax,” are applied. In addition, the applicationClass element, the applicationHash element, the broadcastPermission element, and the appSize element, which are described below, are new items, no previously specified.

14.2. applicationIdentifier element

This element conforms to ARIB STD-B24 Volume 4 “6.2 ApplicationIdentifier element.” Its identifier system shall be integrated with that for broadcast-oriented managed applications. The structure of the applicationIdentifier element is shown in Table 14-5.

Table 14-5 Structure of the applicationIdentifier element

Element/attribute name	Frequency	Definition
applicationIdentifier	-	
orgId	1	Organization identification
appId	1	Application identification

- Semantics
  - ✧ orgId: organization identification. This identifies the organization that has created the application. The identification shall be a unique number.
  - ✧ appId: application identification. This is a number that identifies the application. It shall be unique within the organization identification.

14.3. applicationDescriptor element

This element conforms to ARIB STD-B24 Volume 4 “6.2 applicationDescriptor element.” The structure of the applicationDescriptor element is shown in Table 14-6.

Table 14-6 Structure of the applicationDescriptor element

Element/attribute name	Frequency	Definition

applicationDescriptor		-	
	type	1	Application type
	controlCode	1	Application control code
	visibility	0..1	Visibility
	serviceBound	0..1	Service boundary flag
	priority	1	Application priority
	version	1	Version
	mhpVersion	0..n	MHP version
	icon	0..n	Application icon
	storageCapabilities	0..n	Whether storage can be used

- Semantics
  - ✧ type: application type. This indicates the type of the application that is controlled by an AIT.
  - ✧ controlCode: application control code. This is used to control the application state. For a non-broadcast-oriented managed application, a fixed value, “AUTOSTART” is used.
  - ✧ visibility. Visibility indicates whether this application is visible to the user and other applications while it is being executed.
  - ✧ serviceBound: service boundary flag. This is ignored in non-broadcast-oriented managed applications.
  - ✧ priority: application priority. This indicates the relative priorities between applications in cases where multiple applications operate.
  - ✧ version. The value of this attribute is not specified. The receiver should ignore this value.
  - ✧ mhpVersion: MHP version. For details, refer to ETSI TS 102 809 V1.1.1 (2010-01), Section 5.4.4.8 “MhpVersion.”
    - ✧ profile: application profile. This indicates the application profile of a receiver that can execute this application.
    - ✧ versionMajor: Major version. This indicates the major version of the above profile.
    - ✧ versionMinor: Minor version. This indicates the minor version of the above profile.
    - ✧ versionMicro: Micro version. This indicates the micro version of the above profile.
  - ✧ icon: application icon. For details, refer to ETSI TS 102 809 V1.1.1 (2010-01) Section 5.4.4.6 “IconDescriptor.”
    - ✧ filename: icon file name. This indicates the URL of the icon file.
    - ✧ size: icon size. This indicates the size of the icon data in bytes.
    - ✧ aspectRatio: icon aspect ratio. This indicates the aspect ratio of the icon image. The possible aspect ratios are “4_3,” “16_9,” and “1_1.”
  - ✧ storageCapabilities: This indicates whether storage is available, i.e., whether the application can be retained. “1” means that the application can be retained.

14.4. appName element

Refer to ETSI TS 102 809 V1.1.1 (2010-01) Section 5.4.4.2 “Application.” This element indicates the name of the application.

14.5. applicationLocation element

Refer to ETSI TS 102 809 V1.1.1 (2010-01) Section 5.4.4.23 “SimpleApplicationLocationDescriptorType.”

This element indicates the URL from which the application can be fetched.

14.6. applicationBoundary element

Refer to ETSI TS 102 809 V1.1.1 (2010-01) Section 5.4.4.24 “SimpleApplicationBoundaryDescriptorType.”

The structure of the applicationBoundary element is shown in Table 14-7.

Table 14-7 Structure of the applicationBoundary element

Element/attribute name	Frequency	Definition
applicationBoundary	-	
BoundaryExtension	1..n	Application area

- Semantics
  - ✧ BoundaryExtension: application area. This indicates, in the form of an URL, the valid area from which the application can be fetched. If more than one applicationBoundary element is specified, the area is determining by applying an “OR” operation. The receiver shall not make a transition to an HTML document that is located outside the specified area. Irrespective of the area to which a transition has been specified, if a transition has been made to a general application using the exitFromManagedState() function, all access to broadcast resources is made invalid.

14.7. applicationClass element

This section specifies, as a child element of the Application element, the applicationClass element, which indicates a variety of classes of non-broadcast-oriented managed applications. The structure of the applicationClass element is shown in Table 14-8 and Fig. 14-3 Structure of the applicationClass element. The XML schema of the applicationClass element is shown in Table 14-9.

Table 14-8 Structure of the applicationClass element

Element/attribute name	Frequency	Definition
applicationClass	-	
@layoutClass	0..1	Layout class
@packageClass	0..1	Package class
@certificationClass	0..1	Authentication class

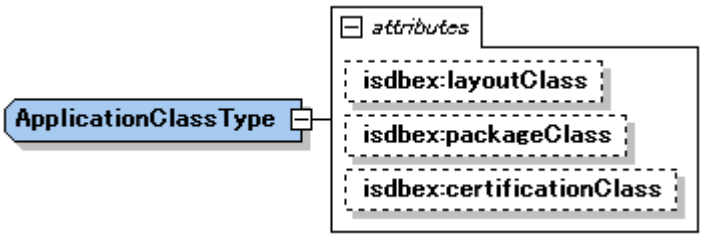


Fig. 14-3 Structure of the applicationClass element

Table 14-9 XML schema of the applicationClass element

```
<xsd:complexType name="ApplicationClassType">
  <xsd:attribute name="layoutClass" type="xsd:string" use="optional"/>
  <xsd:attribute name="packageClass" type="xsd:string" use="optional"/>
  <xsd:attribute name="certificationClass" type="xsd:string" use="optional"/>
</xsd:complexType>
```

- Semantics
  - ✧ layoutClass. This specifies whether the application refers to the broadcast video, and specifies the following three values regarding application display, in accordance with the specification given in 13.2.1.

layoutClass value	Name	Description
compound	Broadcast video inclusion class	Class for an application that refers to the broadcast video as an element and controls its layout in the allotted area.
widget	Widget class	Class for an application that is displayed but that does not refer to

		the broadcast video.
invisible	Non-display class	Class for an application that is not displayed.

✧ packageClass: This specifies the following two values regarding application packaging in accordance with the specification given in 13.2.2.

packageClass value	Name	Description
packaged	Packaged class	Application class in which the entire application is packaged, downloaded and used.
host	Host class	Application class in which the application is fetched as necessary by unit of displayed area and is then used.

✧ certificationClass: This specifies the following two values regarding the application authentication method in accordance with the specification given in 13.5.1.

certificationClass value	Name	Description
method2	Method 2 authentication class	Indicates that Method 2 is used and that Method 2 is not used together with Method 3.
method3	Method 3 authentication class	Indicates that Method 3 is used and that Method 3 may be used together with Method 2.

If the value of this attribute is method2 or method3, authentication using Method 1 can also be used.

Note that, if authentication is to be done using Method 1 alone, it is not necessary to write this attribute.

14.8. broadcastPermission element

This section specifies, as a child element of the Application element, the broadcastPermission

element that declares permission regarding access to a variety of broadcast resources by a non-broadcast-oriented managed application.
The structure of the broadcastPermission element is shown in Table 14-10 and Fig. 14-4. The XML schema of the broadcastPermission element is shown in Table 14-11.

Table 14-10 Structure of a broadcastPermission element

Element/attribute name	Frequency	Definition
broadcastPermission	-	
permissionBitmap	1..n	Access permission bitmap
permissionScope	1	Scope of access permission
@all	0..1	All services flag
scope	0..n	Unit of permission target
@broadcastMedia	0..1	Broadcast media Terrestrial: “terrestrial” BS: ”bs” CS110: ”cs110”
@networkId	0..1	Network ID
@transportStreamId	0..1	Transport stream ID
@serviceId	0..1	Service ID
@affiliationId	0..1	Affiliation ID
@broadcasterId	0..1	Broadcaster ID
@terrestrialBroadcasterId	0...1	Terrestrial broadcaster ID

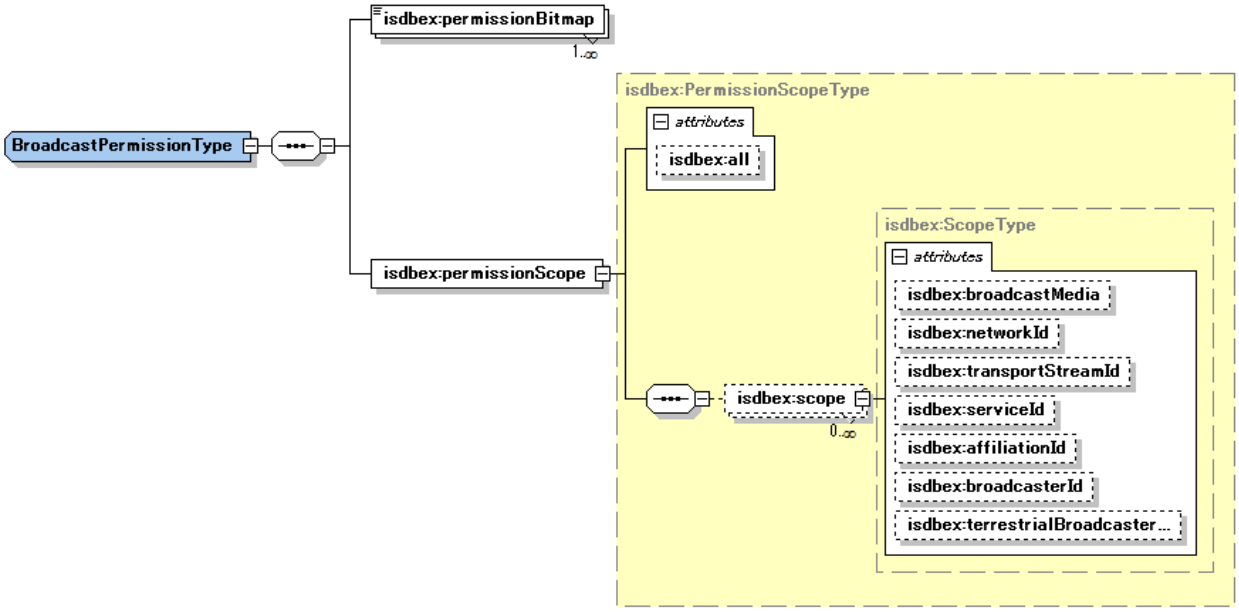


Fig. 14-4 Structure of the broadcastPermission element

Table 14-11 XML schema of the broadcastPermission element

```
<xsd:complexType name="ScopeType">
  <xsd:attribute name="broadcastMedia" type="xsd:string" use="optional"/>
  <xsd:attribute name="networkId" type="xsd:string" use="optional"/>
  <xsd:attribute name="transportStreamId" type="xsd:string" use="optional"/>
  <xsd:attribute name="serviceId" type="xsd:string" use="optional"/>
  <xsd:attribute name="affiliationId" type="xsd:string" use="optional"/>
  <xsd:attribute name="broadcasterId" type="xsd:string" use="optional"/>
  <xsd:attribute name="terrestrialBroadcasterId" type="xsd:string" use="optional"/>
 </xsd:complexType>
 <xsd:complexType name="PermissionScopeType">
  <xsd:sequence>
   <xsd:element name="scope" type="isdbex:ScopeType" minOccurs="0"
        maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="all" type="xsd:boolean" use="optional"/>
 </xsd:complexType>
 <xsd:complexType name="BroadcastPermissionType">
  <xsd:sequence>
```

```
    <xsd:element name="permissionBitmap" type="ipi:Hexadecimal16bit"
          maxOccurs="unbounded"/>
    <xsd:element name="permissionScope" type="isdbex:PermissionScopeType"/>
  </xsd:sequence>
</xsd:complexType>
```

This element specifies the scope of broadcast services in which the given non-broadcast-oriented managed application has permission to access broadcast resources that are indicated in the access permission bitmap. The scope of broadcast services is specified in one or more permission scope units (service, broadcaster, etc.). If there are multiple settings of access permission for an application, multiple broadcastPermission elements are used. If a non-broadcast-oriented managed application being executed attempts to access broadcast resources, and if this application does not have access permission in the broadcast service being received, the receiver shall block this access.

- Semnantics
  - permissionBitmap: access permission bitmap. This shows whether access to each broadcast resource is permitted or not. This is a bitmap of 16 bits, each bit being associated with a specific function. (“1” means that access is permitted.) The first 3 bits identify the bitmap to be used. The assignment of a functional bitmap shall be specified in operational rules.
  - permissionScope: access permission scope. This shows the scope of broadcast services in which access permission is given using the permissionBitmap. The scope of broadcast services is specified using the “all” attribute or the “scope” element, which is a child element.
  - all: all services flag. If the value is “1,” all broadcast services are the target of the above access permission setting, and thus the scope element is ignored.
  - scope: unit of the target of permission. Targets of permission are indicated using (a combination of) the following attributes.

Unit of permission target	Attribute
Broadcast media (terrestrial, BS or CS110)	broadcastMedia
Network	networkId
Transport stream	networkId, transportstreamId
service	networkId, transportstreamId, serviceId
Broadcaster	broadcasterId
Terrestrial broadcaster	terrestrialBroadcasterId

Affiliation	affiliationId

14.9. appSize element

This section specifies, as a child element of the Application element, the appSize element, which indicates the image size for a widget class application. The structure of the appSize element is shown in Table 14-12 and Fig. 14-5. The XML schema of the appSize element is shown in Table 14-13.

Table 14-12 Structure of the appSize element

Element/attribute name	Frequency	Definition
appSize	-	
size	1..n	Size
@range	0..1	Range
@width	1	Width
@height	1	Height

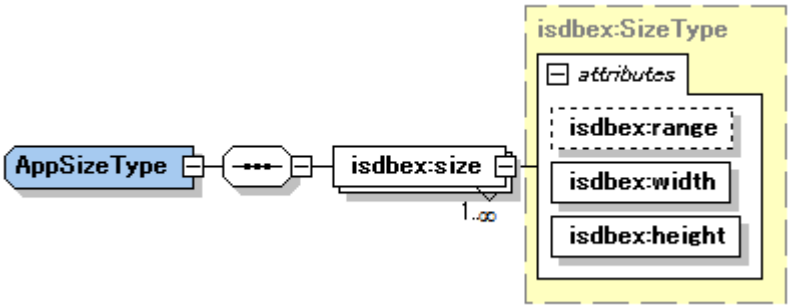


Fig. 14-5 Structure of the appSize element

Table 14-13 XML schema of the appSize element

```
<xsd:complexType name="SizeType">
  <xsd:attribute name="range" type="xsd:string" use="optional"/>
  <xsd:attribute name="width" type="xsd:float" use="required"/>
  <xsd:attribute name="height" type="xsd:float" use="required"/>
 </xsd:complexType>
 <xsd:complexType name="AppSizeType">
```

```
  <xsd:sequence>
    <xsd:element name="size" type="isdbex:SizeType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

This element is used when “widget” is specified in the layoutClass attribute of the applicationClass element. It indicates the size of the application’s rectangular display area by two attributes: width and height. In the case of an application that can set multiple display sizes, either multiple display sizes are specified or the maximum and minimum display sizes are specified.

- Semantics
  - ✧ size: display size. This indicates the display size of an application.
  - ✧ range. If the value of this attribute is “min,” the minimum size of the application is shown. If it is “max,” the maximum size of the application is shown. This attribute is not specified for the normal size.
  - ✧ width. The width of the rectangle that will be superimposed is indicated as a proportion of the width of the entire screen (as a percentage).
  - ✧ height. The height of the rectangle that will be superimposed is indicated as a proportion of the height of the entire screen (as a percentage).

14.10. applicationHash element

This section specifies, as a child element of the Application element, the applicationHash element that stores the application’s hash value. The hash value is encoded in Base64. The method of calculating the application’s hash value shall be specified in operational rules.

14.11. XML schema of the entire AIT

```
<?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:isdbex="urn:arib:isdbex:2014" xmlns:isdb="urn:arib:isdb:2012"
        targetNamespace="urn:arib:isdbex: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:import namespace="urn:arib:isdb:2012"
        schemaLocation="imports/isdb_xmlait.xsd"/>
<xsd:complexType name="SizeType">
 <xsd:attribute name="range" type="xsd:string" use="optional"/>
 <xsd:attribute name="width" type="xsd:float" use="required"/>
 <xsd:attribute name="height" type="xsd:float" use="required"/>
</xsd:complexType>
<xsd:complexType name="AppSizeType">
 <xsd:sequence>
  <xsd:element name="size" type="isdbex:SizeType" maxOccurs="unbounded"/>
 </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ApplicationClassType">
 <xsd:attribute name="layoutClass" type="xsd:string" use="optional"/>
 <xsd:attribute name="packageClass" type="xsd:string" use="optional"/>
 <xsd:attribute name="certificationClass" type="xsd:string" use="optional"/>
</xsd:complexType>
<xsd:complexType name="ScopeType">
 <xsd:attribute name="broadcastMedia" type="xsd:string" use="optional"/>
 <xsd:attribute name="networkId" type="xsd:string" use="optional"/>
 <xsd:attribute name="transportStreamId" type="xsd:string" use="optional"/>
 <xsd:attribute name="serviceId" type="xsd:string" use="optional"/>
 <xsd:attribute name="affiliationId" type="xsd:string" use="optional"/>
 <xsd:attribute name="broadcasterId" type="xsd:string" use="optional"/>
 <xsd:attribute name="terrestrialBroadcasterId" type="xsd:string" use="optional"/>
</xsd:complexType>
<xsd:complexType name="PermissionScopeType">
 <xsd:sequence>
  <xsd:element name="scope" type="isdbex:ScopeType" minOccurs="0"
        maxOccurs="unbounded"/>
 </xsd:sequence>
 <xsd:attribute name="all" type="xsd:boolean" use="optional"/>
</xsd:complexType>
<xsd:complexType name="BroadcastPermissionType">
```

```
  <xsd:sequence>
   <xsd:element name="permissionBitmap" type="ipi:Hexadecimal16bit"
         maxOccurs="unbounded"/>
   <xsd:element name="permissionScope" type="isdbex:PermissionScopeType"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="Application">
  <xsd:sequence>
   <xsd:element name="appName" type="ipi:MultilingualType" minOccurs="0"
         maxOccurs="unbounded"/>
   <xsd:element name="applicationIdentifier" type="isdb:ApplicationIdentifier"/>
   <xsd:element name="applicationDescriptor" type="isdb:ApplicationDescriptor"/>
   <xsd:element name="applicationLocation"
         type="mhp:SimpleApplicationLocationDescriptorType"/>
   <xsd:element name="applicationBoundary"
         type="mhp:SimpleApplicationBoundaryDescriptorType" minOccurs="0"/>
   <xsd:element name="applicationClass" type="isdbex:ApplicationClassType"
         minOccurs="0"/>
   <xsd:element name="applicationHash" type="xsd:base64Binary" minOccurs="0"/>
   <xsd:element name="broadcastPermission" type="isdbex:BroadcastPermissionType"
         minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element name="appSize" type="isdbex:AppSizeType" minOccurs="0"
         maxOccurs="unbounded"/>
  </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="isdbex:Application"
              maxOccurs="unbounded"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:element>
```

```
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
 <xsd:element name="ServiceDiscovery">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:choice>
     <xsd:element name="ApplicationDiscovery"
          type="isdbex:ApplicationOfferingType" maxOccurs="unbounded"/>
    </xsd:choice>
   </xsd:sequence>
   <xsd:attribute name="expirationDate" type="xsd:date" use="required"/>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

14.12. XML signature

The receiver verifies the signature of the AIT if the XML signature specified in this section is allotted to the AIT.

A signature (hereafter called “XML signature”) is allotted to an XML-format AIT in the following way:

A signature is generated for the entire XML-format AIT. Therefore, a single XML signature is allotted to the entire AIT.

The XML signature is specified as follows, based on RFC 3275:

“An XML signature uses an enveloped signature (EnvelopedSignature) that embeds in the document the signature (<Signature> element) for the entire metadata document that is the target of signature.”

14.12.1. <Signature> element

<Signature> element includes the following elements:

- <SignedInfo> element
- <SignatureValue> element
- <KeyInfo> element

14.12.1.1. <SignedInfo> element

<SignedInfo> element includes the following elements:

- <CanonicalizationMethod> element

The value of the "Algorithm" attribute of the <CanonicalizationMethod> element shall be specified in operational rules.

e.g., “http://www.w3.org/TR/2001/REC-xml-c14n-20010315"

- <SignatureMethod> element

The value of the "Algorithm" attribute of the <SignatureMethod> element shall be specified in operational rules.

e.g., “http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/
xmldsig-core-schema.xsd#rsa-sha256"

14.12.1.2. <Reference> element

<Reference> element includes the following elements:

- <Transforms> element

The value of the "Algorithm" attribute of the <Transforms> element shall be specified in operational rules.

e.g., “http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/
xmldsig-core-schema.xsd#enveloped-signature"

- <DigestMethod> element

The value of the "Algorithm" attribute of the <DigestMethod> element shall be specified in operational rules.

e.g., “http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/
xmldsig-core-schema.xsd#sha256"

- <DigestValue> element

A digest value encoded in Base64 shall be stored in the <DigestValue> element.

14.12.1.3. <SignatureValue> element

A signature value encoded in Base64 shall be stored in the <SignatureValue> element.

14.12.1.4. <KeyInfo> element

<KeyInfo> element includes the following elements:

- <KeyData> element

Details of the <KeyData> element shall be specified in operational rules.

Table 14-14 Details of signature element

Element/attribute name	Frequency
Signature	1
SignedInfo	1
CanonicalizationMethod	1
@Algorithm	1
SignatureMethod	1
@Algorithm	1
Reference	1
Transforms	1
@Algorithm	1
DigestMethod	1
@Algorithm	1
DigestValue	1
SignatureValue	1
KeyInfo	1
KeyData	1

Appendix A Extension of Broadcasting Specification

The following extensions to ARIB STD-B24 version 5.7 are necessary to realize the integrated broadcast-broadband services specified in this Specification.

(1) Extension to the broadcast markup language (BML) extension function for broadcasting

- Addition of a function needed to launch an application of an integrated broadcast-broadband service from a data broadcasting service.
- Extension of getBrowserSupport ()
  This is used to determine whether the receiver should launch an integrated broadcast-broadband service.

(2) Addition of the specification of media types

- Addition of the specification of media types for cases where application control information and an application are transmitted via data carousel.

(3) Extension to SI/PSI

- Extension to the data coding method descriptor related to data broadcasting placed in PMT in order to specify the start priorities of data broadcasting and of a broadcast-oriented managed application.
- Extension to the data coding descriptor related to AIT transmission
- Provision of a data coding method identifier related to the transmission of AIT and an application.

(4) Definition of application control information

- Data format of application control table to control application lifecycle

A.1 BML extension function

(1) Addition of the extension function for broadcasting

- startReceivingAIT(): Starts the fetch of application control information

Syntax:

```
Number startReceivingAIT (input String ait_uri)
```

Argument:

ait_uri	URI indicating the source of application control information

Return value:

1:	Success
NaN:	Failure

Description:

When this function is called, the receiver attempts to fetch application control information

from the source specified by ait_uri. This function either returns “Success” as soon as the process of fetching application control information starts, without waiting for the process to complete, and then terminates, or, if it fails to start the fetch process, returns “Failure” and terminates. The receiver holds the completely fetched application control information in the document scope, together with information about its source, and returns its content in response to a getReceivedAIT() call. The number of items of application control information that a receiver can store simultaneously, the number of fetch processes that it can execute in parallel, and the manner in which the receiver operates when it is instructed to fetch more items or processes than these numbers shall be specified in operational rules.

This function does not apply to application control information that is transmitted via section or carousel in the current service and monitored by the receiver.

- getReceivedAIT(): Reads the fetched application control information

Syntax:

```
Array getReceivedAIT ([input String ait_uri])
```

Argument:

ait_uri	URI that indicates the source of application control information to be read. If this is omitted, it is assumed that application control information that is transmitted via section or carousel in the current service and monitored by the receiver has been specified.

Return value:

Array that stores application control information: Success

Array[0]: Control information that is common to both fetch process status and application

Array[0][0]: Application type (Number type)

Array[1]: Control information for the first application

Array[1][0]: Organization identification (Number type)

Array[1][1]: Application identification (Number type)

Array[1][2]: Application control code (Number type)

Array[1][3]: Application entry URL (String type)

・・・・・

Array[n]: Control information for the n-th application

Array[n][0]: Organization identification (Number type)

Array[n][1]: Application identification (Number type)

Array[n][2]: Application control code (Number type)

Array[n][3]: Application entry URL (String type)

0: Fetch of application control information has not been completed, or start of fetch has not been instructed.

-1: Process of fetching application control information has terminated due to error

NaN: Failure

Description:

This function returns the application control information specified by ait_uri as an array if the fetch process has been completed, or returns “0” otherwise. Before calling this function, the fetch of the application control information specified by ait_uri shall be ordered using startReceivingAIT(). However, if the specification of ait_uri is omitted and if application control information monitored by the receiver is to be read, the receiver returns the latest application control information that is being transmitted at the time when this function is called.

Refer to A.4 for details of application type, organization identification, application identification, application control code, and entry URL. The organization identification and application identification included in the return value shall be a character string indicating the value in hexadecimal but “0x” at the head and “h” at the end to indicate hexadecimal expression are removed.

- startAITControlledApp(): Launches an AIT controlled application

Syntax:

```
Number startAITControlledApp(
          input String organization_id
          input String application_id
           [,input String ait_uri])
```

Argument:

organization_id	Character sting indicating the organization identification of the AIT control application to be launched in hexadecimal but “0x” at the head and “h” at the end to indicate hexadecimal expression are removed.
application_id	Character sting indicating the application identification of the AIT control application to be launched in hexadecimal but “0x” at the head and “h” at the end to indicate hexadecimal expression are removed.
ait_uri	URI that includes description of the AIT controlled application to be launched. If this is omitted, it is assumed that the application control information that is transmitted via section or carousel and monitored by the receiver has been specified.

Return value:

1:	Success

NaN: Failure

Description:

When this function is called, the receiver refers to application control information specified by ait_uri. If the application control code of the AIT controlled application specified by organization_id and application_id permits the launch of this AIT control application, the receiver attempts to launch this application. Refer to A.4 for details of application control code.

This function executes the above based on the latest status of the specified application control information regardless of whether or not startReceivingAIT() or getReceivedAIT() has been executed earlier. Therefore, this function may take time because it may fetch application control information.

If the specified application control information cannot be fetched, this function returns “Failure,” and the script execution continues. In other cases, continuation of the execution of the script that follows this function is not guaranteed.

(2) Extension to getBrowserSupport()

・Extension to the extension function group

The following extension function group is added to determine whether the BLM browser supports the launch function for the application execution environment of the integrated broadcast-broadband system. This is specified in additionalinfo when functionname is “APIGroup.”

API	Specification of extension function group (additionalinfo argument)
startReceivingAIT () getReceivedAIT () startAITControlledApp ()	AITControlledApp.Start

・Extension to additionalinfo

The following character string is added to determine whether the receiver supports the application execution environment of the integrated broadcast-broadband system. This is specified in additionalinfo when functionname is “ResidentApp.”

functionname	additionalinfo	Operation of getBrowserSupport()
AITControlledAppEngineFunction	(This shall be specified separately in operational rules)	

A.2 Media types

The following are added to “Annex C Media types,” and “Table 9-6 Format types” in 9.2.1.3 in ARIB STD-B24 Vol. 2.

- XML-format AIT

  Application control information written in XML format specified in A.4.2 of this Specification.

- Section-format AIT

  Application control information in section format specified in A.4.1 of this Specification.

  (This is transmitted via data carousel instead of section.)

- ZIP

  Archive format that combines several files into one file. This is the ZIP file format that is in general use.

  (This is assumed to be used as the application archive format.)

The following values are specified for media type, file type, and format type for each of the above media.

Media name	Media type	File type	Format type
XML format AIT (UTF-16)	application/X-arib-ait+xml; charset=”UTF-16”	0x01	0x432
XML format AIT (UTF-8)	application/X-arib-ait+xml; charset=”UTF-8”	0x01	0x434
Section format AIT	application/X-arib-ait	0x05	0x4a1
ZIP	application/zip	0x05	0x060

A.3 PSI/SI

PSI/SI required when transmitting application control information and an application via broadcast signals is additionally specified in this Specification. The assumed component composition variations are shown below.

(1) Component 1	(2) Component 2	(3) Component 3
Data broadcasting@dc	AIT section	
Data broadcasting@dc	AIT @dc	
Data broadcasting/AIT@dc		
Data broadcasting@dc	AIT section	Application@dc
Data broadcasting@dc	AIT @dc	Application@dc
Data broadcasting@dc	AIT/application@dc	

Data broadcasting/AIT/application@dc		

("@dc" in the table indicates a case where the item concerned is transmitted using the data carousel method specified in ARIB STD-B24 Volume 3.)

The correspondence between the items that need to be additionally specified in PMT in the compositions assumed above and what are specified in the following sections is summarized in the following table.

No.	Target of component identification	A.3.1 Data coding method identification	Additional specification of data coding method descriptor
(1)	Data broadcasting	B24 already specified	A.3.2 Extension to additional_arib_bxml_info()
(2)	AIT	Additionally specified	A.3.3 Specification of ait_identifier_info()
(3)	Application	Additionally specified	None (no domain for additional_data_component_info)

A.3.1 Data coding method identification

Data coding method identification is assigned to each component that transmits application control information or each component that transmits an application based on this Specification.

A.3.2 Extension to the data coding method descriptor related to data broadcasting

The additional_arib_bxml_info() structure of the data coding method descriptor related to data broadcasting placed in PMT in order to specify the start priorities of data broadcasting and broadcast-oriented managed applications shall be extended as follows.

Data structure	Number of bits	Bit string notation
additional_arib_bxml_info(){		
transmission_format	2	bslbf
entry_point_flag	1	bslbf
if(entry_point_flag= = 1){		
auto_start_flag	1	bslbf
document_resolution	4	bslbf
use_xml	1	bslbf
default_version_flag	1	bslbf
independent_flag	1	bslbf

style_for_tv_flag	1	bslbf
reserved_future_use	4	bslbf
if(default_version_flag= = 0){		
bml_major_version	16	uimsbf
bml_minor_version	16	uimsbf
if(use_xml= = 1){		
bxml_major_version	16	uimsbf
bxml_minor_version	16	uimsbf
}		
}		
}else{		
reserved_future_use	5	bslbf
}		
If(transmission_format= = '00'){		
additional_arib_carousel_info()		
ondemand_retrieval_flag	1	bslbf
file_storable_flag	1	bslbf
start_priority	1	bslbf
reserved_future_use	5	bslbf
}else if(transmission_format= = '01'){		
reserved_future_use	8	bslbf
}		
}		

The semantics of the extension parameter is as follows:

start_priority: Indicates the start priority of BML/B-XML content when BML/B-XML content and an AIT controlled application are transmitted simultaneously. When this field is 0, the start control of BML/B-XML content shall comply with ARIB STD-B24 Volume 4 Paragraph 5.1.

Value	Semantics
0	Priority of the BML/B-XML content concerned is not specified
1	BML/B-XML content is launched with high priority regardless of whether an AIT controlled application is present or not

A.3.3 Extension to data coding method descriptor related to AIT

The following ait_identifier_info(), which is specified in ARIB STD-B24, is used for additional_data_component_info in the data coding method descriptor in the PMT related to AIT in order to apply it to AIT used for the control of a broadcast-oriented managed application. However,

the number of loops within the descriptor is limited to 1 in this Specification.

Data structure	Number of bits	Bit string notation
ait_identifier_info(){		
for (i= 0;i<N;i++){		
application_type	16	uimsbf
transport_type	1	bslbf
application_priority	2	baslbf
AIT_version_number	5	uimsbf
}		
}		

The semantics of the parameters is as follows:

application_type: Indicates the value of the application type transmitted in AIT. The value for HTML5 applications is 0x010 in this Specification.

transport_type: indicates AIT transmission method.

Value	Semantics
0	Data carousel transmission if the XML-format AIT file
1	AIT section transmission

application_priority: Indicates start priority order for application types that are subject to control by AIT. This is 00 if no priority is specified.

AIT_version_number: Indicates the version_number shown in AIT

A.4 Formats of application control information

A.4.1 Section format AIT

The application information table (AIT) that controls broadcast-oriented managed applications is specified in the section format.

A.4.1.1 AIT section structure

Data structure	Number of bits	Bit string notation
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_descriptor_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
}		

Semantics:

table_id: This shall be set to 0x74.

section_syntax_indicator: Always 1.

section_length: Indicates the number of bytes from the section length field to the end of the section containing CRC32. This shall not exceed 1021 (0x3FD in hexadecimal).

application_type: Indicates the application type that is subject to control by AIT. The value for HTML5 applications is 0x0010 in this Specification.

version_number: Indicates the version number of the sub-table. The version number increments by one each time information within the sub-table is changed. When this value becomes 31, the next incremented number returns to 0.

current_next_indicator: Always 1.

section_number: Indicates the section number. The section number of the first section in the sub-

table is 0x00. The section number increments by one each time a section with the same table identification and application type is added.

last_section_number: Indicates the section number of the last section in the sub-table.

common_descriptor_length (common descriptor loop length): Indicates the byte length of the common descriptor domain that follows. Descriptors within this descriptor domain are applied to all applications within the AIT sub-table.

application_control_code: Indicates the control code that is used to control the application state. The following values are specified in this Specification.

Value	Identification name	Semantics
0x01	AUTOSTART	Launches the application
0x02	PRESENT	Indicates that the application is operable
0x04	KILL	Terminates the application
0x05	PREFETCH	Fetches and holds the application

application_loop_length (application information loop length): Indicates the byte length of the entire loop that includes application information that follows

application_identifier(): Value that uniquely identifies the application. See A.4.1.2for details.

application_descriptors_loop_length (application information descriptor loop length): Indicates the byte length of the application information descriptor domain that follows. Descriptors within this descriptor domain are applied to only the compliant application.

A.4.1.2 Application identification

An application is uniquely identified using the following application identifier. This identifier is a 6-byte (48 bit)-length structure and is stored in AIT.

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

Semantics:

organization_id: Indicates the organization that has created the application. This ID shall be unique worldwide.

application_id: Indicates the number that identifies the application. This ID shall be unique within the organization ID.

A.4.1.3 Application descriptor

One application descriptor shall be placed in the application information descriptor loop in AIT for each application.

Data structure	Number of bits	Bit string notation
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	baslbf
reserved_future_use	5	baslbf
application_priority	8	uimsbf
for (j= 0;i<N;i++){		
transport_protocol_label	8	uimsbf
}		
}		

Semantics:

descriptor_tag: This shall be set to 0x00.

application_profiles_length (application profile information length): Indicates the byte length of the entire application profile information included in the loop that follows. The number of loops is limited to 1 in this Specification.

application_profile: Indicates the application profile of the receiver on which this application can run. If the receiver has this profile, it is capable of executing this application. The content of the profile is defined for each application type. In this Specification, this is made up of bitmaps, each defined for each function required for the receiver. The top 3 bits indicate switching of the bitmap. A bitmap is specified for each version. The assignment of function bitmaps shall be specified in operational rules. The top 3 bits are fixed at 000.

version.major (major version): Indicates the major version of the above profile. This is fixed at 1 in

this Specification.

version.minor (minor version): Indicates the minor version of the above profile. This is fixed at 1 in this Specification.

version.micro (micro-version): Indicates the micro version of the above profile. This is fixed at 1 in this Specification.

The minimum profile to execute this application is indicated in the above four fields. The receiver launches this application if a profile that satisfies at least one of the following logical operations (i.e., the logical operation result is “true”) exists within this application profile information.

(Application profile ∈ A set of profiles implemented on the device)
AND { (application major version < major version of the device that implements this profile)
OR [(application major version = major version of the device that implements this profile)
AND ({application minor version < minor version of the device that implements this profile}
OR { [application minor version = minor version of the device that implements this profile]
AND [application micro version ≦ micro version of the device that implements this profile]})] }

service_bound_flag (service boundary flag): Indicates whether this application is only valid in the current service. If this flag is 1, the application is linked to only the current service. If the service is switched to another service, the application terminates. This is fixed at 1 in this Specification.

visibility: Indicates whether this application is visible to the user and other applications while it is being executed. This is fixed at 11 in this Specification. (This value means that the application is visible to the user and other applications.)

application_priority: Indicates the relative priorities between applications in cases where multiple applications operate. This is fixed at 0xFF in this Specification, indicating that the application priority is not specified.

transport_protocol_label: Indicates the value that uniquely identifies the application transmission method. This corresponds to the field of the same name in the transport protocol descriptor.

A.4.1.4 Transport protocol descriptor

The purpose of the transport protocol descriptor is to specify the transport protocol used in broadcast, communication, etc. to transmit the application and to indicate the location of the application, which depends on the transport protocol used.

There shall be as many transport protocol descriptors as the number of the transport protocol labels in the application descriptor in the common descriptor loop or the application information descriptor loop in AIT.

Data structure	Number	Bit string

	of bits	notation
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
}		
}		

Semantics:

descriptor_tag: This shall be set to 0x02

protocol_id: Indicates the protocol used to transmit the application.

Value	Semantics
0x0000	reserved_future_use
0x0001…0x0002	reserved
0x0003	HTTP/HTTPS transmission
0x0004	data carousel transmission
0x0005…0xFFFF	reserved future_use

transport_protocol_label: Indicates a value that uniquely identifies the application transmission method. This corresponds to the field of the same name in the application descriptor.

selector_byte (selector domain): Stores supplementary information specified for each protocol ID. The data structures for HTTP/HTTPS transmission and data carousel transmission are shown below.

- Selector domain in the case of HTTP/HTTPS transmission

Data structure	Number of bits	Bit string notation
for (i= 0;i<N;i++){		
URL_base_length	8	uimsbf
for (j= 0;j<URL_base_length;j++){		
URL_base_byte	8	uimsbf
}		
URL_extension_count	8	uimsbf
for (j= 0;j<URL_extension_count;j++){		
URL_extension_length	8	uimsbf
for (k= 0;k<URL_extension_length;k++){		

URL_extension_byte	8	uimsbf
}		
}		
}		

Semantics:

URL_base_length: Indicates the number of bytes in the URL base for fetching the application.

URL_base_byte (URL base): Character string of the URL base for fetching the application.

URL_extension_count: Indicates the number of URL extensions for fetching the application. The presence of multiple URL extensions means that there are multiple locations into which the application can be fetched, within a single URL base domain.

URL_extension_length: Indicates the number of bytes of the URL extension for fetching the application.

URL_extension_byte (URL extension): Character string of the URL extension for fetching the application. The fact that the entirety takes on a loop structure means that multiple URL base domains that can fetch the application can be set.

- Selector domain in the case of data carousel transmission

Data structure	Number of bits	Bit string notation
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

Semantics:

remote_connection: 1 is specified when AIT and the data carousel that transmits the application are not transmitted in the same service.

original_network_id: If remote_connection is 1, sets the identification of the original network in which the application is transmitted.

transport_stream_id: If remote_connection is 1, sets the identification of the transport stream in which the application is transmitted.

service_id: If remote_connection is 1, sets the identification of the service in which the application is transmitted.

component_tag: Sets the component tag value that indicates the service component in which the

application is transmitted.

A.4.1.5 Simple application location descriptor

The purpose of the simple application location descriptor is to specify details of the application source. One simple application location descriptor shall be placed in the application information descriptor loop in AIT for each application.

Data structure	Number of bits	Bit string notation
simple_application_location_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for (i= 0;i<N;i++){		
initial_path_bytes	8	uimsbf
}		
}		

Semantics:

descriptor_tag: This shall be set to 0x15.

initial_path_bytes (application URL): Character string that indicates the URL of the entry document of the relevant application. This is expressed as a relative path to the location from which the application specified in the transport protocol descriptor can be fetched.

A.4.1.6 Application boundary and permission descriptor

The purpose of the application boundary and permission descriptor is to set the application boundary and the broadcast resource access permission for each domain (URL). One or more application boundary and permission descriptors are placed in the application information descriptor loop in AIT. If this descriptor is not in place, the application boundary is unlimited and any access to broadcast resources is permitted.

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

permission_bitmap_count		
for (j= 0;j<permission_bitmap_count;j++){	16	bslbf
permission_bitmap		
}	8	uimsbf
managed_URL_count		
for (j= 0;j<managed_URL_count;j++){	8	uimsbf
managed_URL_length		
for (k= 0;k<managed_URL_length;k++){	8	uimsbf
managed_URL_byte		
}		
}		
}		
}		

Semantics:

descriptor_tag: This shall be set to 0x30.

permission_bitmap_count: Indicates the number of specified permission_bitmaps. How to specify multiple bitmaps shall be specified in operational rules.

permission_bitmap (access permission bitmap): Information about whether access to each broadcast resource is expressed as bitmaps, each of which is defined for each function. The upper 3 bits indicate bitmap switching. The assignment of function bitmaps shall be specified in operational rules.

managed_URL_count (access permission managed domain setting count): Indicates the number of domain settings to which access permission setting given in the permission_bitmap is applied. If this value is 0, access permission setting is applied to all domains. In other words, it is understood that the URL can be any arbitrary location. However, if a specific domain has been specified as the setting for another access permission bitmap, that setting is effective. (This is based on the rule that the smaller domain access permission setting takes precedence.)

managed_URL_length (access permission managed domain setting byte length): Indicates the number of bytes in the access permission management domain setting (the character string of the URL).

managed_URL_byte (access permission managed domain setting information): Indicates the character string of the URL of the access permission management domain. The domain or a subdirectory in it is specified.

A.4.1.7 Autostart priority descriptor

The purpose of the autostart priority descriptor is to specify application start priorities. At most, one start priority information descriptor is placed in the application information descriptor loop in AIT for

each application. This descriptor shall be placed only in the application information description in which the application control code orders automatic start (AUTOSTART, etc.) of the application. If this descriptor is not in place, the priority of the application, including data broadcasting, can be assumed to be the lowest.

Data structure	Number of bits	Bit string notation
autostart_priority_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
autostart_priority	8	uimsbf
}		

Semantics

descriptor_tag: This shall be set to 0x31.

autostart_priority (start priority order): Indicates the start priority order of the relevant application among the data broadcasting associated with the service currently being received and all broadcast-oriented managed applications. The possible value is limited to 1 or 2 in this Specification. Value 1 indicates that the application has the highest priority, while value 2 indicates that data broadcasting has the highest priority.

A.4.1.8 Cache control information descriptor

The cache control information descriptor is used for cache control when the resource that constitutes the application is to be cached in a case where reuse of the application is assumed. At most one cache information descriptor is placed in the application information descriptor loop in AIT for each application.

If this descriptor is not in place, the receiver deletes the resources that constitute the application at the time when the application terminates.

Data structure	Number of bits	Bit string notation
cache_control_info_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
application_size	16	uimsbf
cache_priority	8	uimsbf
package_flag	1	bslbf
application_version	7	uimsbf

expire_date	16	bslbf
}		

Semantics:

descriptor_tag: This shall be set to 0x32.

application_size: The size of the entire application is shown in kbytes. Value 0 is to be used if the size is unknown.

cache_priority: Indicates the priority of holding the application in cache. It is assumed that the larger this value is, the higher the priority. It is assumed that if the application size exceeds the cache capacity, the cache is emptied, application by application, using this information as reference information. Value 0xFF is to be used if no priority is to be specified.

package_flag: Indicates whether the application has been packaged into a single file. Value 1 indicates that the application has been packaged in this way.

application_version: Indicates the application version number. The receiver memorizes the application version of the cached application. Afterwards, if the application version has been updated at the time of application launch, the receiver fetches a new version of the application from the specified URL, uses it, and replaces the cache content, instead of using the version of the application stored in the cache.

expire_date (cache expiration date): Indicates the application cache expiration year, month and date in the lower 16 bits of the MJD. The receiver may retain the cache until this expiration date. After the expiration date, the cache content is deleted. Value 0xFFFF is specified if there is no expiration date.

A.4.1.9 Randomized latency descriptor

The purpose of the randomized latency descriptor is to delay the application control by a stochastically set delay time in order to distribute the load of access to servers to fetch the application. At most one randomized latency descriptor is placed in the application information descriptor loop in AIT for each application. If this descriptor is not in place, the control specified in the control code is executed at the time when a specific version of AIT is received for the first time.

Data structure	Number of bits	Bit string notation
randomized_latency_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
range	16	uimsbf
rate	8	uimsbf
randomization_end_time_flag	1	bslbf
reserved_future_use	7	bslbf

if(randomization_end_time_flag= = 1){
randomization_end_time
}
}
	40	bslbf

Semantics:

descriptor_tag: This shall be set to 0x33.

range (range of delay): Indicates the maximum delay time between the current time and the application of the control code. This is specified in seconds.

rate (number of delay settings): Indicates the number of stages in the stochastically set delay time before the control code that has been set is applied. The receiver calculates the delay time Td using the formula of Td= N×range÷rate, where N is a randomly selected integer between 0 and rate, and delays the application of the control code by Td from the time AIT was received.

randomization_end_time_flag (stochastically applied end time flag): Indicates whether stochastically applied end time (randomization_end_time) is specified. Value 1 indicates that the end time is specified.

randomization_end_time (stochastically applied end time): Expiration time of the stochastically applied delay process. If AIT is received after this time, the control code is applied immediately. Date is coded in the lower 16 bits of MJD, and the hour, minute, and second in Japan Standard Time (JST) are coded in BCD 24-bit.

A.4.2 XML-format AIT

XML-format AIT complies with ETSI TS 102 809 (V1.1.1) DVB Signalling and carriage of interactive applications and services in hybrid broadcast/broadband environments Section 5.4 “XML-based syntax,” with some extensions. Hereafter, the above specification is simply referred to as “the ETSI Specification” for simplicity. Figure A-1 shows the upper-level structure of XML format AIT.

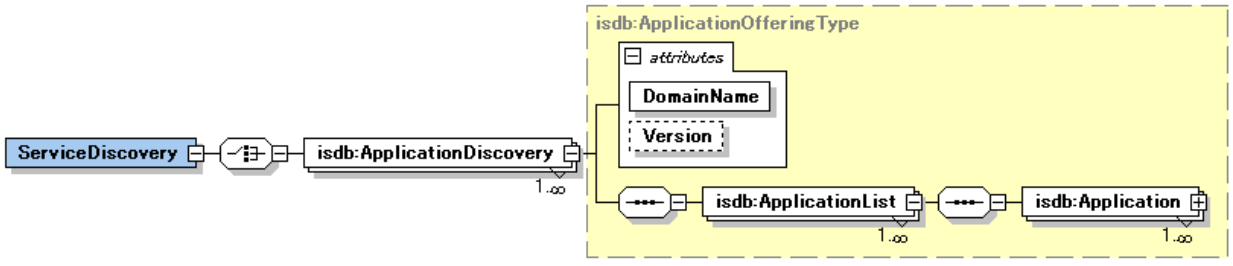


Figure A-1 Upper-level structure of XML-format AIT

The ServiceDisovery element is on the top, followed by the ApplicationDiscovery element, which is further followed by the ApplicationList element. These elements comply with the ETSI Specification. However, since the application elements below these and the elements that further follow it are

specified additionally as below, the name space of each upper element is specified as isdb:ApplicationDiscovery and isdb:ApplicationList.

A.4.2.1 Application element

The following XML schema is applied additionally to the application element. The application element specified in the ETSI Specification Section 5.4.4.2 is not applied. This element corresponds to part of the AIT section structure (part of the application loop) in A.4.1.1. The semantics of each information element is also common to that in A.4.1.1.

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

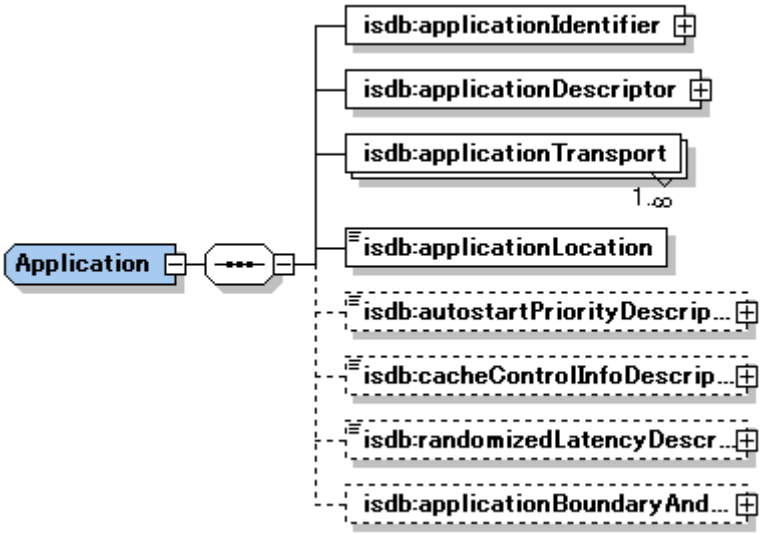


Figure A-2 Structure of the application element

A.4.2.2 ApplicationDescriptor element

The following XML schema is applied additionally as the ApplicationDescriptor element. The ApplicationDescriptor element specified in the ETSI Specification 5.4.4.4 and the ApplicationType element specified in 5.4.4.11 are not applied. This element corresponds to part of the AIT section structure in A.4.1.1 and A.4.1.2 (part of the application loop) and the application descriptor in A.4.1.3. The semantics of each information element is also common to these. ISDB-HTML is introduced as the ApplicationType element used in this Specification.

```
<xsd:simpleType name= "IsdbApplicationType">
    <xsd:restriction base= "xsd:string">
        <xsd:enumeration value= "ISDB-HTML"/>
    </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name= "ApplicationType">
    <xsd:choice>
        <xsd:element name= "IsdbApp" type= "isdb:IsdbApplicationType"/>
        <xsd:element name= "OtherApp" type= "mpeg7:mimeType"/>
    </xsd:choice>
</xsd:complexType>
<xsd:complexType name= "ApplicationDescriptor">
    <xsd:sequence>
        <xsd:element name= "type" type= "isdb: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>
```

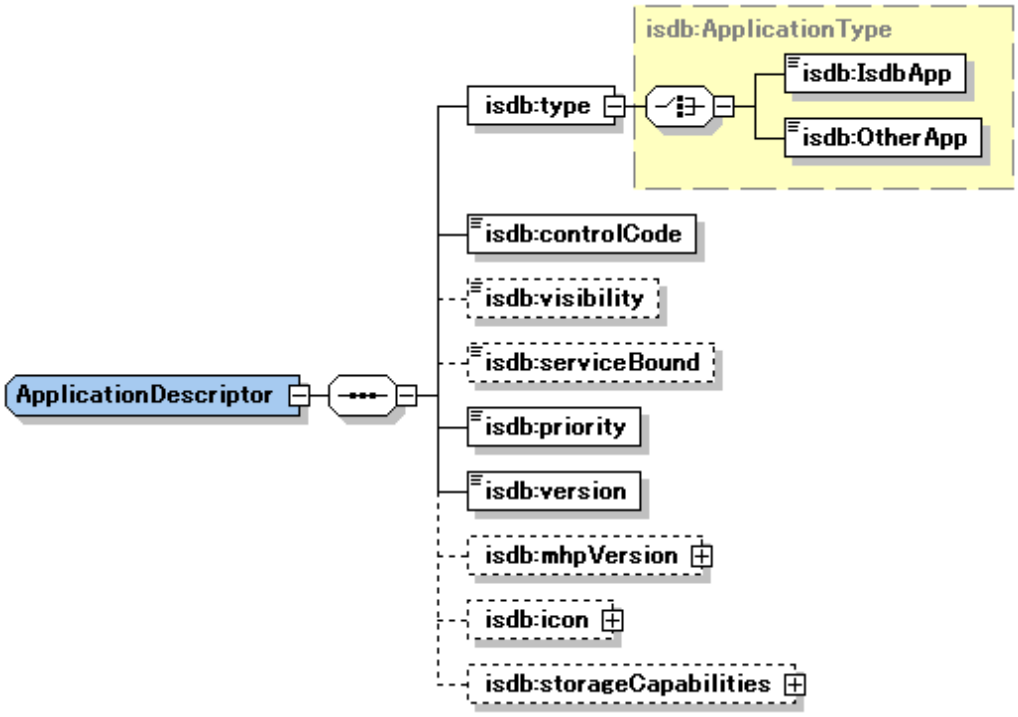


Figure A-3 Structure of the ApplicationDescriptor element

A.4.2.3 ApplicationTransport element

The following XML schema is applied additionally. This element corresponds to the transport protocol descriptor in A.4.1.4. The semantics of each information element is also the same as for that descriptor.

```
<xsd:complexType name= "HTTPTransportType">
    <xsd:complexContent>
```

```
        <xsd:extension base= "isdb: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= "isdb:TransportProtocolDescriptorType">
            <xsd:sequence>
                <xsd:element name= "DvbTriplet" type= "ipi:DVBTriplet"/>
                <xsd:element name= "ComponentTag" type= "isdb:ComponentTagType"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name= "TransportProtocolDescriptorType" abstract= "true"/>
```

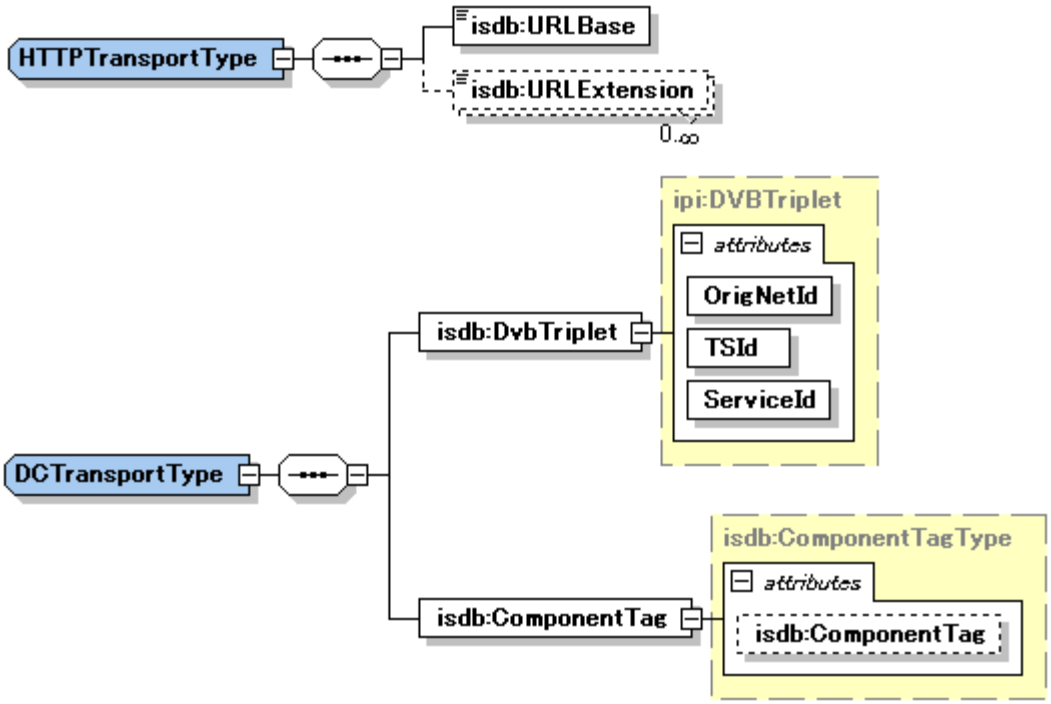


Figure A-4 Structure of the ApplicationTransport element

A.4.2.4 ApplicationBoundaryAndPermissionDescriptor element

The following XML schema is applied additionally as the ApplicationBoundaryAndPermissionDescriptor element. This element corresponds to the application boundary and permission descriptor in A.4.1.6. The semantics of each information element is also the same as for that descriptor.

```
<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= "isdb:BoundaryAndPermissionType"
            maxOccurs= "unbounded"/>
    </xsd:sequence>
```

```
</xsd:complexType>
```

Figure A-5 Structure of the applicationBoundaryAndPermissionDescriptor element

A.4.2.5 AutostartPriorityDescriptor element

The following XML schema is applied additionally. This element corresponds to the autostart priority descriptor in A.4.1.7. The semantics of each information element is also the same as for that descriptor.

```
<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>
```

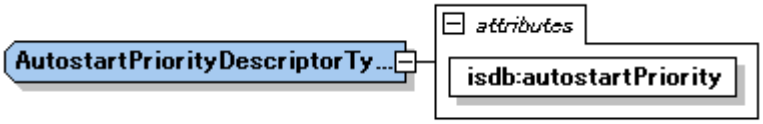


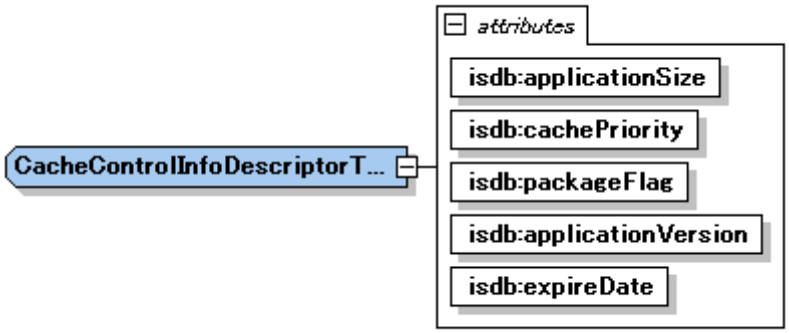
Figure A-6 Structure of the autostartPriorityDescriptor element

A.4.2.6 CacheControlInfoDescriptor element

The following XML schema is applied additionally as the CacheControlInfoDescriptor element. This element corresponds to the cache control information descriptor in A.4.1.8. The semantics of each information element is also the same as for that descriptor.

```
<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>
```



FigureA-7 Structure of the cacheControlInfoDescriptor element

A.4.2.7 RandomizedLatencyDescriptor element

The following XML schema is applied additionally as the RandomizedLatencyDescriptor element. This element corresponds to the randomized latency descriptor in A.4.1.9. The semantics of each information element is also the same as for that descriptor.

```
<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>
```

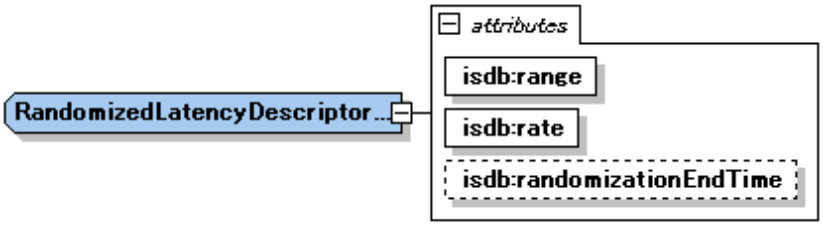


Figure A-8 Structure of the randomizedLatencyDescriptor element

A.4.2.8 XML schema of the overall XML format AIT

The XML schema of the entire XML format AIT is shown below.

```
<?xml version= "1.0" encoding= "UTF-8"?>
<isdb:ServiceDiscovery
xmlns:isdb= "urn:arib:isdb:2012"
xmlns:ipi= "urn:dvb:metadata:iptv:sdns:2008-1"
xmlns:mhp= "urn:dvb:mhp:2009"
xmlns:tva= "urn:tva:metadata:2005"
xmlns:mpeg7= "urn:tva:mpeg7:2005"
xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation=
"urn:arib:isdb:2012 isdb_xmlait.xsd">
<isdb:ApplicationDiscovery DomainName= "test.com">
<isdb:ApplicationList>
<isdb:Application>
<isdb:applicationIdentifier>
<mhp:orgId>19</mhp:orgId>
<mhp:appId>1</mhp:appId>
</isdb:applicationIdentifier>
<isdb:applicationDescriptor>
<isdb:type>
<isdb:IsdbApp>ISDB-HTML</isdb:IsdbApp>
</isdb:type>
<isdb:controlCode>AUTOSTART</isdb:controlCode>
<isdb:priority>1a</isdb:priority>
<isdb:version>1a</isdb:version>
</isdb:applicationDescriptor>
<isdb:applicationTransport xsi:type= "isdb:HTTPTransportType">
<isdb:URLBase>http:///www.xbc.co.jp</isdb:URLBase>
```

```
      </isdb:applicationTransport>
      <isdb:applicationLocation>http://www.xbc.co.jp/a.html</isdb:applicationLocation>
        <isdb:autostartPriorityDescriptor isdb:autostartPriority= "1"/>
        <isdb:cacheControlInfoDescriptor isdb:cachePriority= "80"
       isdb:applicationSize= "134" isdb:expireDate= "2012-12-12"
       isdb:packageFlag= "false" isdb:applicationVersion= "1"/>
        <isdb:randomizedLatencyDescriptor
       isdb:rate= "60" isdb:range= "60" isdb:randomizationEndTime= "2012-11-11"/>
        <isdb:applicationBoundaryAndPermissionDescriptor>
        <isdb:boundaryAndPermission>
          <isdb:permissionBitmap>1ffe</isdb:permissionBitmap>
          <isdb:managedURL>http:///www.xbc.co.jp</isdb:managedURL>
        </isdb:boundaryAndPermission>
        <isdb:boundaryAndPermission>
          <isdb:permissionBitmap>1fff</isdb:permissionBitmap>
          <isdb:managedURL>http:///www.abc.co.jp</isdb:managedURL>
          <isdb:managedURL>http://www.xyz.com</isdb:managedURL>
        </isdb:boundaryAndPermission>
        </isdb:applicationBoundaryAndPermissionDescriptor>
       </isdb:Application>
     </isdb:ApplicationList>
   </isdb:ApplicationDiscovery>
</isdb:ServiceDiscovery>
```

A.4.2.9 Example of XML format AIT statements

```
<?xml version= "1.0" encoding= "UTF-8"?>
<isdb:ServiceDiscovery
 xmlns:isdb= "urn:arib:isdb:2012"
 xmlns:ipi= "urn:dvb:metadata:iptv:sdns:2008-1"
 xmlns:mhp= "urn:dvb:mhp:2009"
 xmlns:tva= "urn:tva:metadata:2005"
 xmlns:mpeg7= "urn:tva:mpeg7:2005"
 xmlns:xsi= ”http://www.w3.org/2001/XMLSchema-instance” xsi:schemaLocation= "urn:arib:isdb:2012
 isdb_xmlait.xsd">
   <isdb:ApplicationDiscovery DomainName= "test.com">
```

```
<isdb:ApplicationList>
    <isdb:Application>
        <isdb:applicationIdentifier>
            <mhp:orgId>19</mhp:orgId>
            <mhp:appId>1</mhp:appId>
        </isdb:applicationIdentifier>
        <isdb:applicationDescriptor>
            <isdb:type>
                <isdb:IsdbApp>ISDB-HTML</isdb:IsdbApp>
            </isdb:type>
            <isdb:controlCode>AUTOSTART</isdb:controlCode>
            <isdb:priority>1a</isdb:priority>
            <isdb:version>1a</isdb:version>
        </isdb:applicationDescriptor>
        <isdb:applicationTransport xsi:type= "isdb:HTTPTransportType">
            <isdb:URLBase>http:///www.xbc.co.jp</isdb:URLBase>
        </isdb:applicationTransport>
        <isdb:applicationLocation>http://www.xbc.co.jp/a.html</isdb:applicationLocation>
        <isdb:autostartPriorityDescriptor isdb:autostartPriority= "1"/>
        <isdb:cacheControlInfoDescriptor isdb:cachePriority= "80"
         isdb:applicationSize= "134" isdb:expireDate= "2012-12-12"
         isdb:packageFlag= "false" isdb:applicationVersion= "1"/>
        <isdb:randomizedLatencyDescriptor
         isdb:rate= "60" isdb:range= "60" isdb:randomizationEndTime= "2012-11-11"/>
        <isdb:applicationBoundaryAndPermissionDescriptor>
            <isdb:boundaryAndPermission>
                <isdb:permissionBitmap>1ffe</isdb:permissionBitmap>
                <isdb:managedURL>http:///www.xbc.co.jp</isdb:managedURL>
            </isdb:boundaryAndPermission>
            <isdb:boundaryAndPermission>
                <isdb:permissionBitmap>1fff</isdb:permissionBitmap>
                <isdb:managedURL>http:///www.abc.co.jp</isdb:managedURL>
                <isdb:managedURL>http://www.xyz.com</isdb:managedURL>
            </isdb:boundaryAndPermission>
        </isdb:applicationBoundaryAndPermissionDescriptor>
    </isdb:Application>
</isdb:ApplicationList>
```

```
    </isdb:ApplicationDiscovery>
</isdb:ServiceDiscovery>
```

Appendix B Conversion into/out of Other Character Codes

Conversion from shifted-JIS, EUC-JP, and the eight-level code into UCS shall comply with ARIB STD-B24 Vol. 1 Part 2 Annex F.

Appendix C Use Case of Receiver Operation

C.1 Basic scenario for a broadcast-oriented managed application

The basic scenario for a broadcast-oriented managed application, including its launch and termination, is first described.

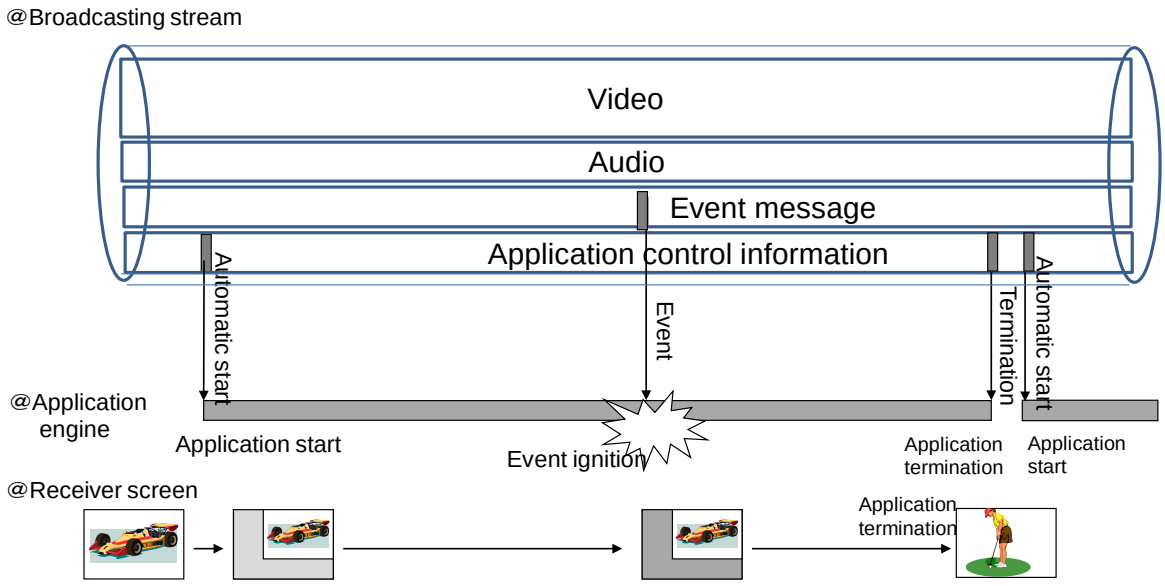


Figure C-1 Basic scenario for a broadcast-oriented managed application

- Assumed scenario

  While the receiver is receiving a broadcast service, an associated application is automatically launched at the start of a specific program. Both the broadcast video and the application are displayed simultaneously. After this, the display of the application switches in synchronization with changes in the scenes in the program. When the program ends, the application is terminated and the receiver displays only the broadcast video. The application control information that controls the application operation is inserted and transmitted in the broadcast stream.

- Operation sequence

  (1) When the program associated with an application starts while the receiver is receiving a specific broadcast service, application control information begins to be transmitted. The

receiver detects the application control information being transmitted, receives it, monitors it, and fetches new information every time it is updated.

(2) If the application control code of the application control information fetched at the start of the program indicates AUTOSTART, the receiver checks whether the application can be run on it. If it can, the receiver accesses the application URL, fetches (the start page of) the application, and instructs the application engine to launch it. The receiver operates in accordance with the application description, and displays both the broadcast video and the application simultaneously.

(3) If what is displayed is to be switched in synchronization with a certain scene in the middle of the program, the application transmits the corresponding event message at that time. The receiver receives it and notifies the application engine of it via the application control part. The application has an event description associated with the event message ID. When this event ignites (i.e., arrives), what is displayed is switched as specified.

(4) Since it is assumed that the user may make channel selection in the middle of the program, application control information that indicates automatic start is transmitted repeatedly throughout the program. The receiver constantly detects any update of application control information, checks that the application in operation is either AUTOSTART or PRESENT, and terminates the application due to time-out if neither is specified.

(5) At the time when the program ends, application control information is updated. If this information specifies the application in operation and instructs the receiver to terminate (KILL) it, the application control part of the receiver instructs the application engine to terminate the specified application. Thereupon, the application is terminated. After this, the receiver refers to PMT, checks the start priorities of data broadcasting and the application, and operates accordingly.

(6) If another application is associated with a new program, the application control information is updated. The updated application control information specifies AUTOSTART of the new application when the program is switched. When the receiver receives this application control information, it terminates the application that is associated with the previous program and is still in operation, and returns to process (2) above, and fetches and launches the new application.

C.2 Scenario for simultaneous operation of data broadcasting and a broadcast-oriented managed application

This section assumes simultaneous operation of data broadcasting and a broadcast-oriented managed application. It considers various assumed operation scenarios and the operations of receivers of various levels of compliance to this Specification.

- Assumed scenario variations

Operation scenario variations are classified as follows:

- ✓ Case 0: Data broadcasting only
- ✓ Case 1: Application only
- ✓ Case 2: Data broadcasting + application (automatic start) with application given higher priority
- ✓ Case 3: Data broadcasting + application (automatic start) with data broadcasting given higher priority
- ✓ Case 4: Data broadcasting + application (no automatic start)

Variations of the receiver' level of compliance are classified as follows.

- ✓ Type0: Supports only data broadcasting
- ✓ Type1: Supports both data broadcasting and applications
- ✓ Type2: Supports only applications

Assumed operation of each type of receiver in each operation case is summarized in Table C-1.

Table C-1 Patterns of simultaneous operation with data broadcasting, and receiver types

Case	Type 0 receiver	Type 1 receiver	Type 2 receiver
Case 0	Data broadcasting is launched automatically or with a press on the d button	Data broadcasting is launched automatically or with a press on the d button	Data broadcasting does not operate
Case 1	Data broadcasting does not operate	Data broadcasting is launched automatically	Data broadcasting is launched automatically
Case 2	Data broadcasting is launched automatically or with a press on the d button	Data broadcasting is launched automatically	Data broadcasting is launched automatically
Case 3	Data broadcasting is launched automatically or with a press on the d button	Data broadcasting is launched (application may be launched afterwards)	Data broadcasting is launched automatically
Case 4	Data broadcasting is launched automatically or with a press on the d button	Data broadcasting is launched automatically (application may be launched afterwards)	Data broadcasting does not operate automatically (application is launched by user

			operation)

Whether the application is launched automatically is determined based on whether an application with AUTOSTATRT specified in the application control code of application control information is present or not. In Cases 2 and 3, this depends on the start priority setting.

Simultaneous operation of data broadcasting and an application occurs in Cases 2 to 4. So, detailed operation scenarios of Cases 2 to 4 are described below.

- Operation scenario for Case 2

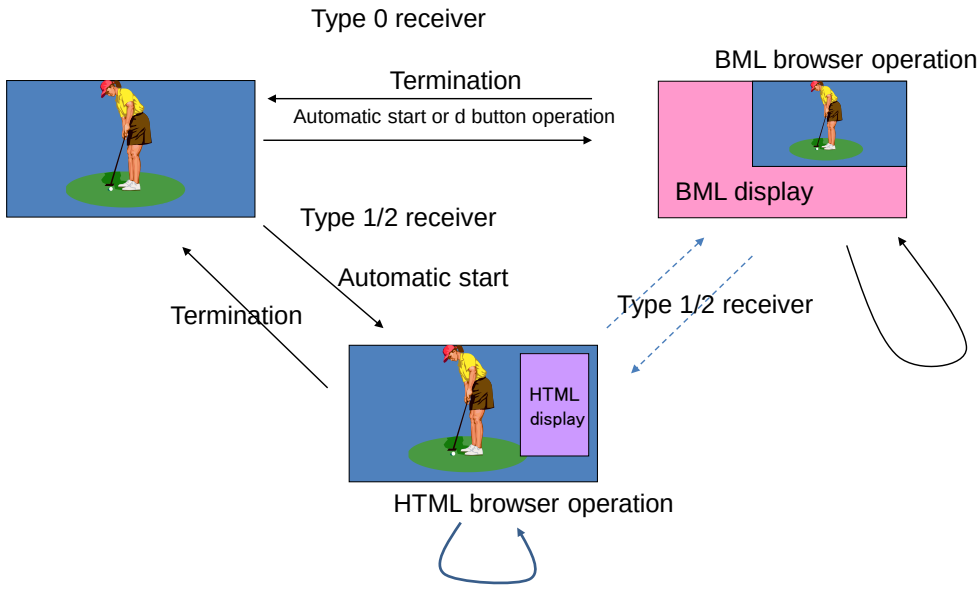


Figure C-2 Example of receiver display screen transitions in Case 2

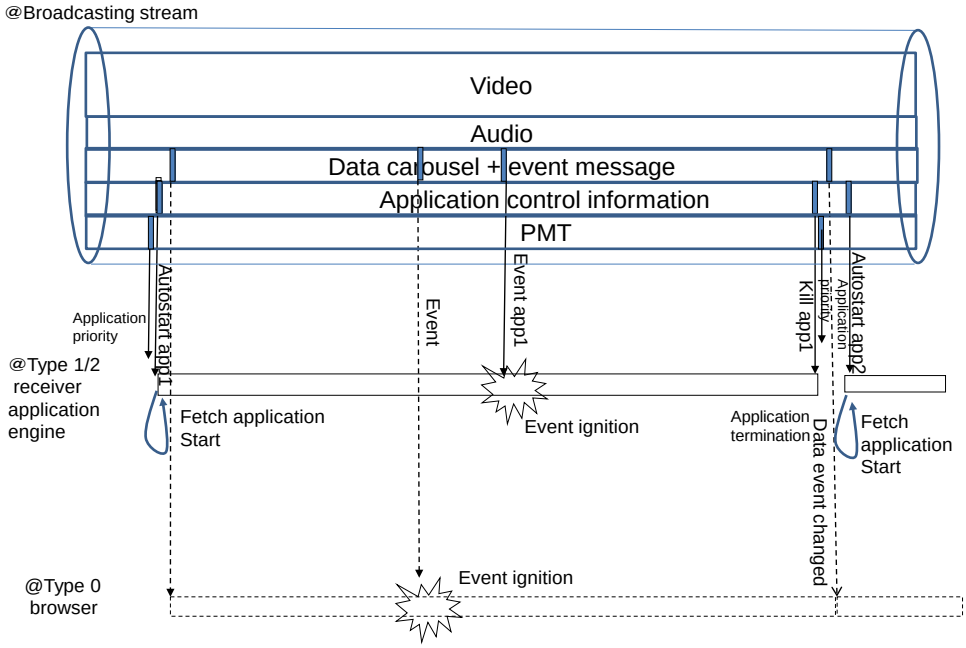


Figure C-3 Receiver operation in Case 2

A) Operation of a Type 1 receiver

(1) The receiver checks the start priority specified in PMT at the time of channel selection. If it confirms that the priority of the application is high, it receives application control information. If this indicates AUTOSTART, it fetches and executes the application.

(2) If the receiver receives an event message in the middle of the ongoing program, and confirms that it is an event message for this application, it instructs the application engine to process the event.

(3) When the receiver receives application control information that orders application termination at the time of program termination, it terminates the application at that time. After the application termination, it refers to PMT and checks the application system priority. If it confirms that the priority of the application is high, it waits for an update of application control information.

(4) If the receiver receives application control information indicating AUTOSTART of a new application at the start of the next program, it fetches and executes the specified new application. After this, processes (2)-(4) are repeated.

B) Type 2 receiver operation

The same as Type 1 except that the start priority is not checked.

- Operation scenario for Case 3

A) Operation of a Type 1 receiver

(1) The receiver checks the start priority specified in PMT at the time of channel selection. If this confirms that the priority of data broadcasting is high, it executes the data broadcasting reception process. If this is AUTOSTART, it fetches the data broadcasting startup document and starts its execution.

(2) For example, in processing the startup document, the receiver checks the receiver capability. If it can execute the application and if it is connected to the Internet, it displays the entry button to the application service. (In the case of a Type 0 receiver, the display of the button is suppressed on the basis of the receiver capability check.) When the user operates this button, the receiver calls the application launch function from data broadcasting, and terminates data broadcasting. Using this function call, the receiver checks the application control information fetched immediately before. If it confirms that the application is operable (i.e., the application is in either the “Enabled” or the “Active” state), it fetches and launches the application. (If the receiver fetches application control information in XML-format AIT from the server in accordance with what is specified in the above function, it fetches and starts the application in accordance with this control information.)

(3) If, at the time of program termination, the receiver receives application control information ordering application termination, it terminates the application. The receiver then refers to PMT after the application termination and checks the application priority. If this confirms that the priority of data broadcasting is high, it fetches the data broadcasting startup document and begins to execute it. After this, processes (2)-(3) are repeated.

B) Operation of a Type 2 receiver

(1) When the receiver recognizes that an application is available at the time of channel selection, it receives application control information. If this is AUTOSTART, it fetches and executes the application.

(2) If it receives an event message in the middle of the ongoing program, and confirms that it is an event message for this application, it instructs the application engine to process the event.

(3) If, at the time of program termination, the receiver receives application control information that orders termination, it terminates the application.

(4) The receiver receives updated application control information at the start of the next program. If this indicates AUTOSTART, it fetches and executes the specified new application. After this, processes (2)-(4) are repeated.

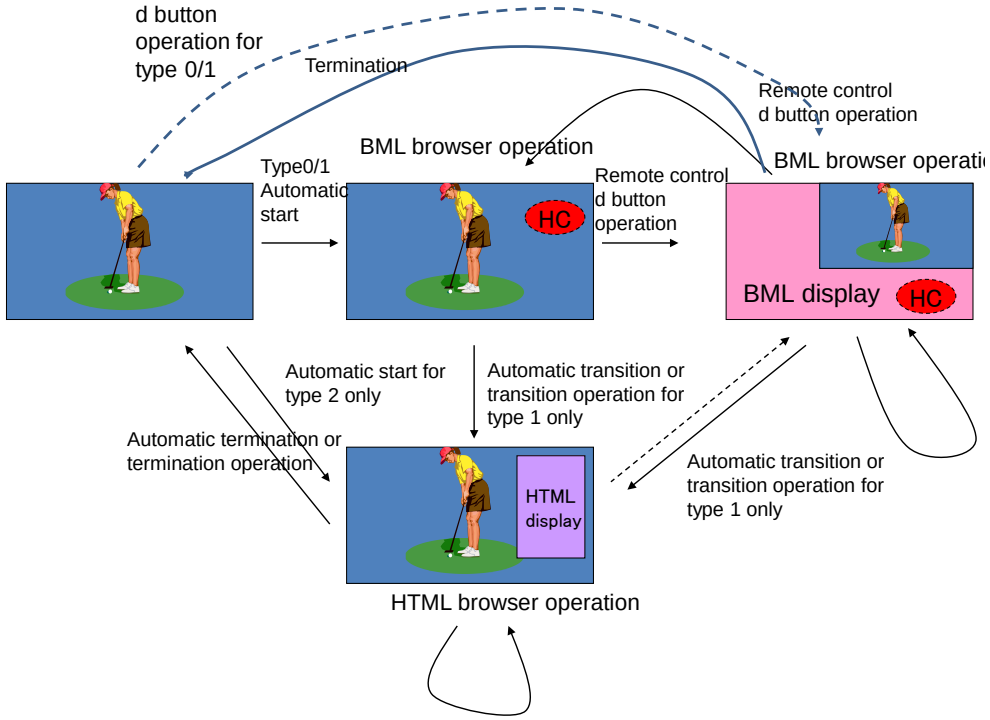


Figure C-4 Example of receiver display screen transitions in Case 3

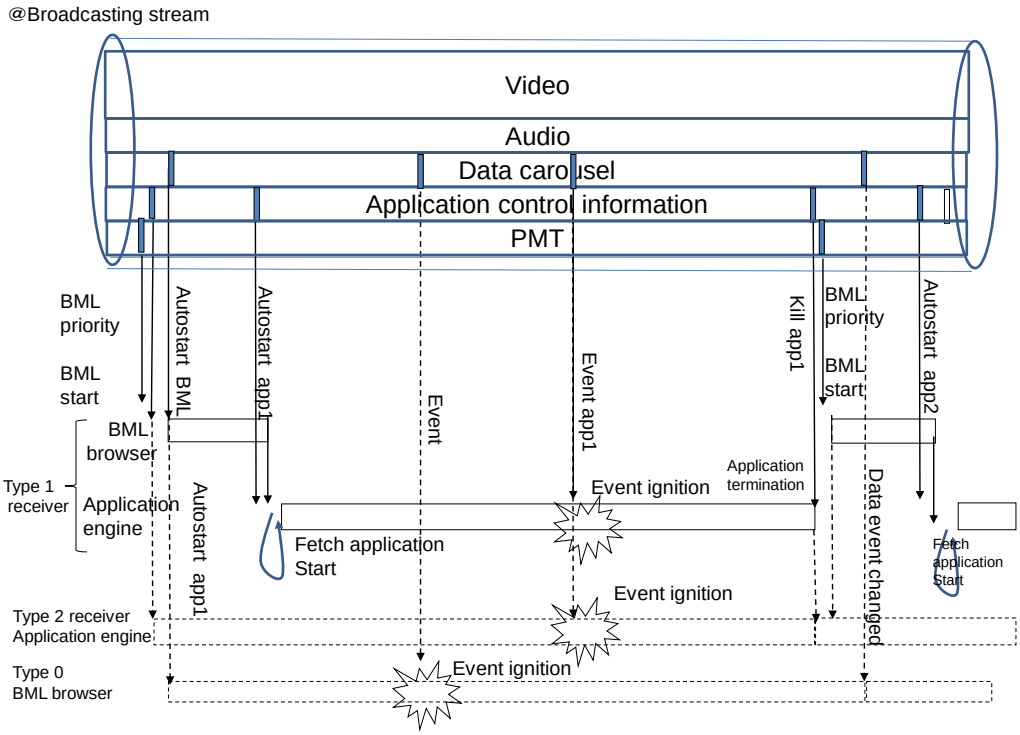


Figure C-5 Receiver operation in Case 3

- Operation scenario in Case 4 (the case where application control information is transmitted via broadcast signals)

  In this case, application control information can either be transmitted via broadcast signals, or be fetched from the server via communication. In either case, the display screen transitions shown in Figure C-6 can be assumed.

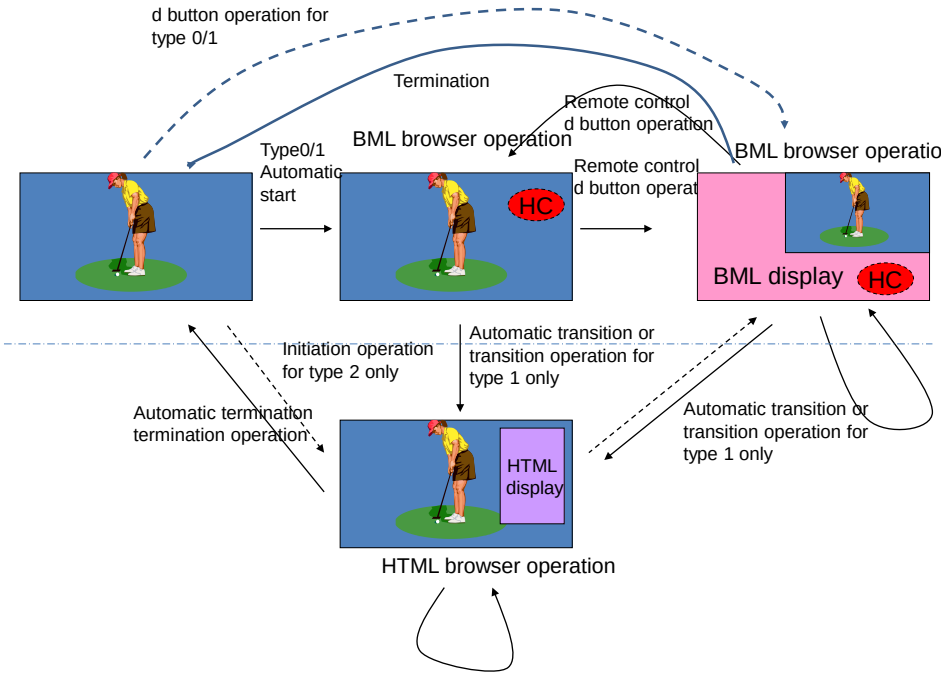


Figure C-6 Example of receiver display screen transitions in Case 4

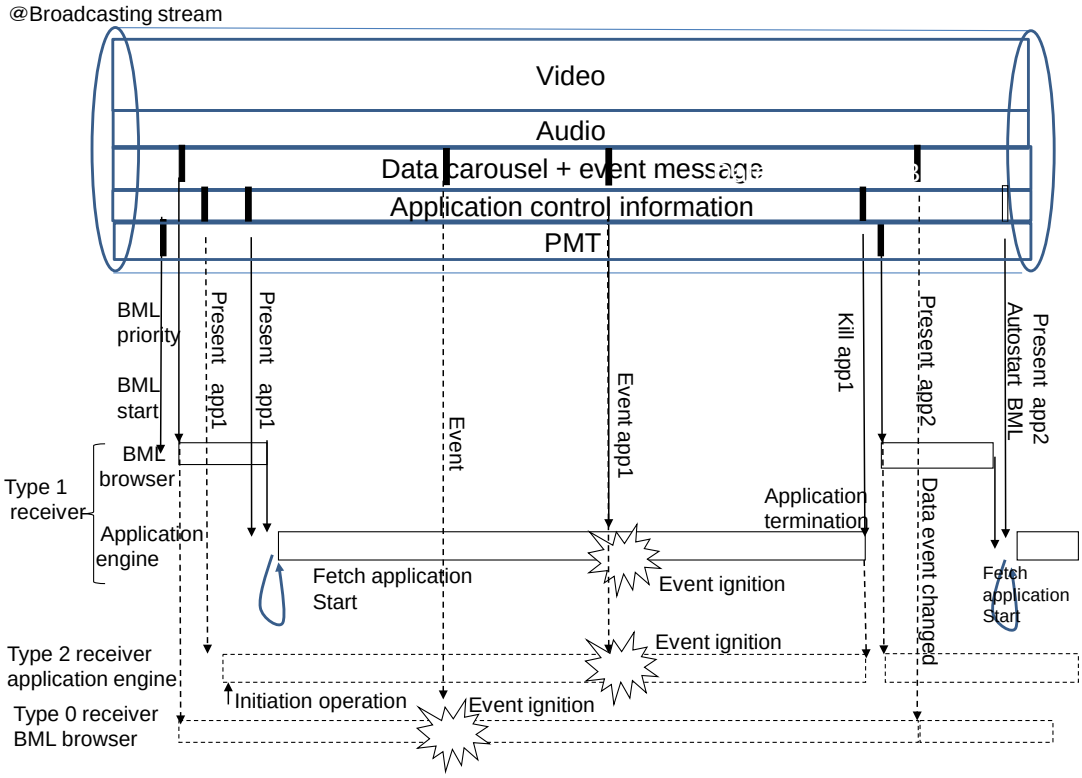


Figure C-7 Receiver operation in Case 4 (transmission of application control information via broadcast signals)

A) Operation of a Type 1 receiver

(1) The receiver checks the start priority specified in PMT at the time of channel selection. If this confirms that the priority of data broadcasting is high, it executes the process of receiving data broadcasting. If this is AUTOSTART, it fetches the data broadcasting startup document and begins to execute it.

(2) For example, in processing the startup document, the receiver checks the receiver capability. If it can execute the application and if it is connected to the Internet, it displays the entry button to the application service. (In the case of a Type 0 receiver, the display of the button is suppressed on the basis of the receiver capability check.) When the user operates this button, the receiver calls the application launch function from data broadcasting, and terminates data broadcasting. Using this function call, the receiver checks the application control information fetched immediately before. If it confirms that the application is operable (i.e., the application is in either the “Enabled” or the “Active” state), it fetches and launches the application.

(3) If, at the time of program termination, the receiver receives application control information ordering application termination, it terminates the application. It refers to PMT after the application termination and checks the application priority. If it confirms that the priority of

data broadcasting is high, it fetches the data broadcasting startup document and begins to execute it. After this, processes (2)-(3) are repeated.

B) Type 2 receiver operation

A type 2 receiver cannot start any application because there is no application set to start automatically. (However, it will be technically possible to start the operation of an application if the Specification is revised in the future in the following way. When application control information indicates that one or more applications are operable, the receiver displays a means by which the user can select an application from these and start it. When the user performs an operation to start an application, that application is started.)

- Operation scenario for Case 4 (In the case where application control information is fetched from the server)

This is the only scenario that works when application control information is not fetched via broadcast signals but received as a file. In this case, data broadcasting is always launched, and this in turn launches the application.

A) Operation of a Type 1 receiver

(1) The receiver checks the start priority specified in PMT at the time of channel selection. If this confirms that the priority of data broadcasting is high, it executes the data broadcasting reception process. If this is AUTOSTART, it fetches the data broadcasting startup document and starts its execution.

(2) For example, in processing the startup document, the receiver checks the receiver capability. If it can execute the application and if it is connected to the Internet, it displays the entry button for the application service. (In the case of a Type 0 receiver, the display of the button is suppressed on the basis of the receiver capability check.) When the user operates this button, the receiver calls the application launch function from data broadcasting, and terminates data broadcasting. The receiver fetches an application control information file from the URL specified in the above function call, analyzes it, fetches from the server only the one application for which automatic start is specified, and starts it.

(3) If the application is to be terminated at the time of program termination, the receiver terminates the application at the appropriate time, which is determined either using a timer after checking the current time, or using a means such as an event message. After the application has terminated, the receiver refers to PMT to check the application priority. If it recognizes that the priority of data broadcasting is high, it fetches the data broadcasting startup document and starts its execution. After this, processes (2)-(3) are repeated.

B) Operation of a Type 2 receiver

The Type 2 receiver has no means of accessing application control information on the server. Therefore, it cannot start an application.

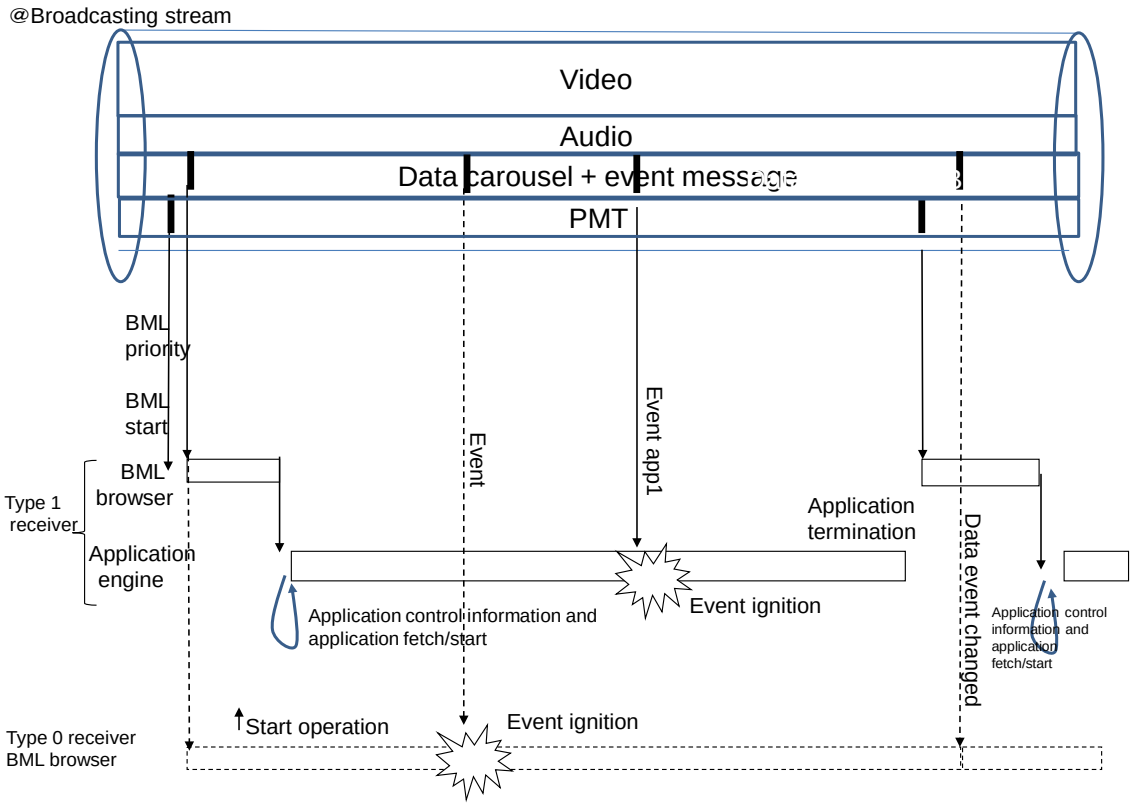


Figure C-8 Receiver operation in Case 4 (application control information is fetched from the server)

C.3 Scenario for multiple broadcast-oriented managed applications

- Assumed scenario

  In this case, multiple applications are associated with broadcasting. The following scenario can be conceived as an example. There are applications that are created by different providers. They are grouped according to their purpose, such as news headlines or weather forecast. They have different security levels. They are reusable and are distinguished using application identifiers. An entry application that makes up a menu window for selecting one of these applications is automatically started either at the start of the program or at the time of channel selection. When the user selects one application, a transition is made to the application. In this case, the destination application of the transition is also a broadcast-oriented managed application, and the display of the broadcast video continues. When the program ends, the application in operation, whatever it is, is terminated.

- Operation sequence

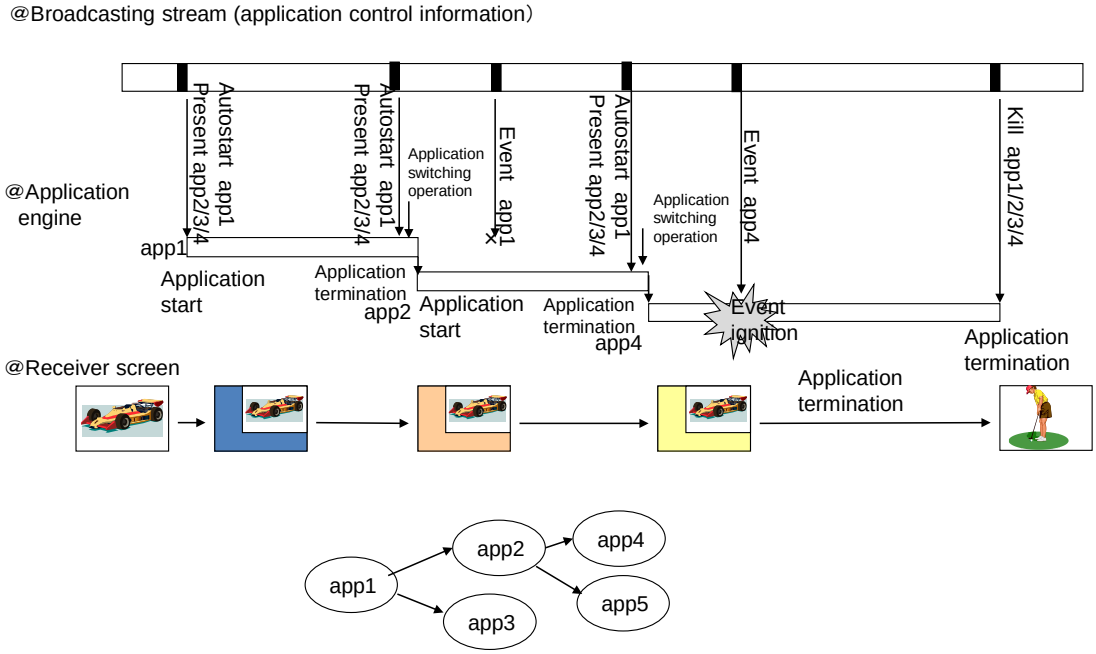


Figure C-9 Receiver operation in the multi-application scenario

(1) The receiver fetches application control information when the program starts, and starts app1, for which automatic start is specified.

(2) While app1 is in operation, the user selects another application. This causes a function call to order a transition to another application (app2). The receiver checks whether app2 specified in the application control information that was fetched immediately before is operable, and if it is, the receiver fetches the specified new application and executes it. This means, from the application control perspective, that app1 is terminated and app2 is started.

(3) While app2 is in operation, no response is made to any event message that is meant for the other application (app1).

(4) Furthermore, a transition is later made from app2 to app4 on user instruction in the manner described in (2).

(5) If an event message for app4 is received while app4 is in operation, this event ignites in the application engine, and, the screen is switched to the one specified in the application.

(6) If, at the time of program termination, the receiver receives application control information ordering the termination of app4, it terminates the application. If any of app1 through app4 that has been selected by the user and remains in operation is to be discontinued at the time of program termination, an application control code specifying all the applications (app1

through app4) shall be contained in the application control information.

C.4 VOD play scenario in a broadcast-oriented managed application

- Assumed scenario

  In this scenario, VOD streaming is launched from an application associated with the broadcast program, and what is displayed is first switched from the broadcast video to the VOD video, and is later switched back to the broadcast video.

- Operation sequence

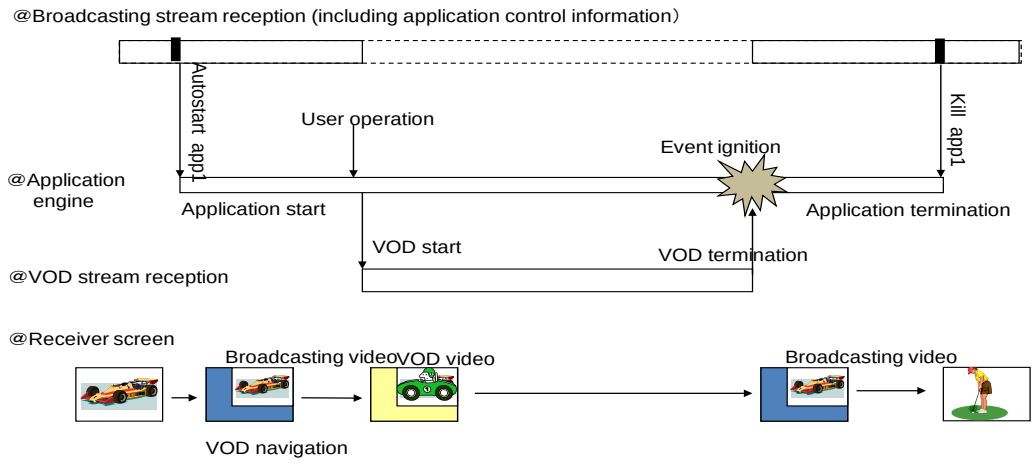


Figure C-10 Receiver operation in the VOD playback scenario

(1) The receiver fetches application control information when the program starts, and launches app1, for which automatic start is specified.

(2) While app1 is in operation, the user selects the VOD service. This makes the receiver start receiving VOD streaming. At this point, the application switches from broadcast video display to VOD display.

(3) To continue the execution of the broadcast-oriented managed application when the VOD is being received and played, it is necessary to continue receiving broadcasting. However, this may not be possible depending on the receiver implementation. If it is not possible to receive and play both broadcasting and VOD simultaneously, the reception and playing of broadcasting may be discontinued.

(4) When the VOD ends, the application receives an event message indicating the termination of the VOD play, and switches from VOD display to broadcast video display. If the receiver

discontinued the reception and playing of broadcasting in (3), it re-launches the reception of broadcasting.

(5) When the program ends, the receiver receives application control information ordering the termination of app1, and terminates the application.

C.5 Companion device-coordinated service scenario in a broadcast-oriented managed application

- Assumed scenario

It is assumed in this scenario that the user installs, in a programmable mobile device such as a smartphone or a tablet, an application that provides a companion device-coordinated function. This function includes discovering a receiver, exchanging data with the receiver, etc. Alternatively, receiver manufacturers may provide such a device in the form of a sophisticated remote unit. Hereafter, the above application is called a companion application. The user launches a companion application while watching a broadcast service. At the instruction from a broadcast-oriented managed application that runs on the receiver, the companion application launches the specified program-associated application on the mobile device. After this, in synchronization with changes in the program scene, both the display of the broadcast-oriented managed application on the receiver and the display of the companion application on the mobile device are switched. The user's operations on the mobile device can switch not only the display of the companion application on the mobile device but also the display of the broadcast-oriented managed application on the receiver. When the program ends, both the broadcast-oriented managed application on the receiver and the program-associated companion application on the mobile device are terminated.

Figures C-11, C-12, and C-13 show examples of screen transitions, of the configuration of software on the receiver and the mobile device, and of the service sequence in the companion device-coordinated scenario.

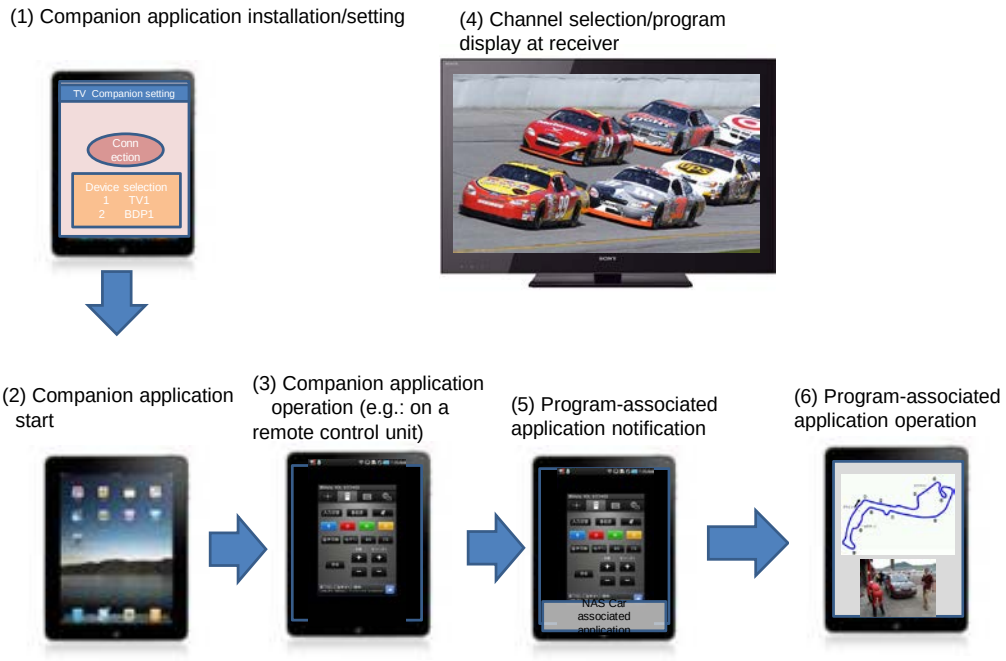


Figure C-11 Example of screen transitions in a companion device-coordinated service

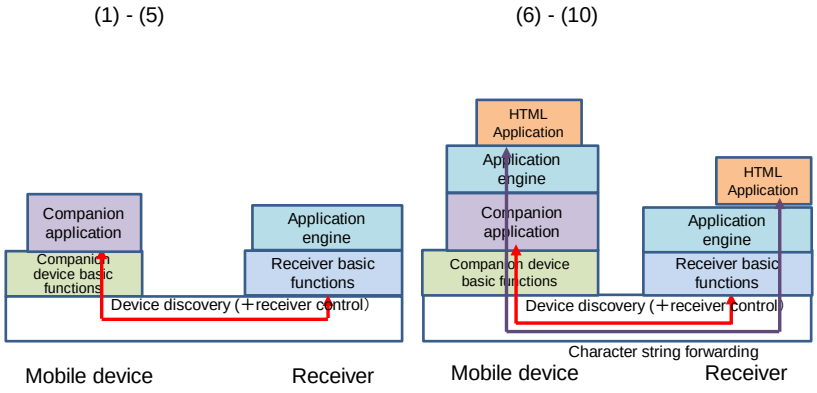


Figure C-12 Software structure example for receivers and mobile devices

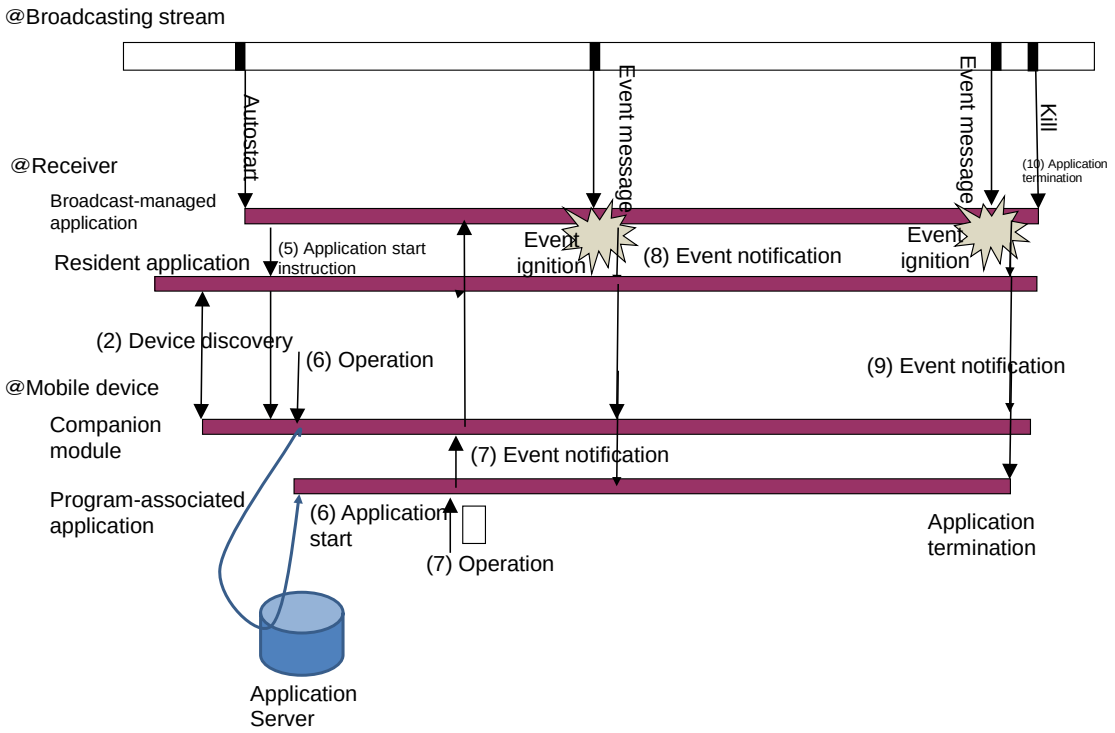


Figure C-13 Example of a companion device-coordinated service sequence

➢ Operation sequence

(1) The user installs a companion application on the mobile device. The communication sequence for device discovery is set as the default. In this sequence, the receiver with which the application is to be associated is searched for and registered.

(2) The user launches the companion application. This initiates the execution of the device discovery communication sequence. The companion application recognizes the receiver and becomes able to communicate with it.

(3) The user begins to use the companion application. (Example: The user uses the companion device as a remote control unit and controls the receiver to select the desired broadcast service.)

(4) The receiver selects the specified broadcast service to start to play video/audio, and launches the broadcast-oriented managed application. (This may be executed before (2) and (3).)

(5) The broadcast-oriented managed application on the receiver executes a function that launches the program-associated application that runs on an external device. The instruction to launch the application is communicated to the companion application via the receiver basic functions. The companion application on the mobile device displays the presence of

the program-associated application on the mobile device screen and displays a message asking for the user's approval.

(6) In response to this message, the user operates the mobile device to give approval. As a result, the companion application fetches the program-associated application from the specified URL, and launches it.

(7) If the user performs some sort of operation while the program-associated application on the mobile device is in operation, the application display on the mobile device is switched accordingly. At the same time, the program-associated application executes a function that generates and forwards a character string that gives notification of the user operation event. This character string is forwarded to the broadcast-oriented managed application on the receiver via the companion application of the mobile device. Consequently, the display of the broadcast-oriented managed application on the receiver is also switched.

(8) When the receiver receives the event message included in broadcast signals and the broadcast-oriented managed application recognizes this, the screen displayed on the receiver is switched. This event also makes the broadcast-oriented managed application execute a function that forwards a character string that gives notification of the specific event. This character string is forwarded to the program-associated application on the mobile device via the receiver basic functions and the companion application on the mobile device. Consequently, the screen displayed by the program-associated application on the mobile device is also switched.

(9) When the program ends, the receiver receives an event message. The application on the receiver orders the termination of the application on the external device in the same manner as in (8). As a result, the program-associated application on the mobile device is terminated.

(10) When the program ends, the receiver receives application control information that orders the termination of the application, and terminates the broadcast-oriented managed application.

C.6 Document transition scenario of a broadcast-oriented managed application that contains an iframe

This document transition scenario takes into consideration the application boundary and control of access to broadcast resources in the case where a broadcast-oriented managed application uses an iframe element.

- **Assumed scenario**

  Even if a broadcast-oriented managed application contains an iframe, it is possible to control access of all HTML documents, including documents within the iframe, to broadcast resources based on the application boundary and access permission setting information (application boundary and permission descriptor in AIT) included in the application control information. As

long as the broadcaster creates and manages the “parent” HTML document that constitutes a broadcast-oriented managed application that contains an iframe element, there would be no need to worry about the application boundary and access permission. However, when it comes to “children” HTML documents that are referred to in the iframe element, it is necessary to consider the following two cases:

A) Access permission is provided to the children documents on the condition that a certain level of management is carried out. However, transitions to outside of a defined domain need to be prohibited.

B) Freedom of transition is given using ordinary web pages. However, no permission is given to access broadcast resources.

The scenarios for the receiver operation based on the application control information settings for these opposite extreme cases are described below.

- **Scenario for Case A**

Figure C-14 shows the concept of this case.

(1) After the receiver has started to receive a broadcast service, it fetches and analyzes application control information, and holds the application boundary and access permission settings (application boundary and permission descriptor in AIT). Three Internet domains are set in the control information with the following application boundary. Domain-A is a broadcaster domain, in which the application entry document is placed and in which access to all broadcast resources is permitted. Domain-B is a non-broadcaster domain, where access to NVRAM is not allowed. Domain-C is another non-broadcaster domain, where access to SI is not allowed.

(2) The receiver fetches the entry document of a broadcast-oriented managed application from Domain-A based on the application URL specified in the application control information, and executes it. Since access to all broadcast resources is allowed in Domain-A, the entry document is allowed to use various functions, such as referring to broadcast video, referring to SI, and accessing NVRAM. Therefore, the broadcast video can be displayed in a sub-screen, and buttons for a transition to Document a1, Document b1, or Document c1 can also be displayed.

(3) When the user operates button a, the receiver fetches Document a1 from the same domain, Domain-A, and makes a transitions to the document. This document is also given all permissions just like the entry document. In the display of Document a1, the broadcast video continues to be displayed in a sub-screen, and Document a2, which is referred to in the iframe element, is displayed in a rectangular sub-screen inserted in the screen of Document a1. Since Document a2 is fetched from Domain-B, it is denied access to NVRAM. If Document a2 has a statement that orders access to NVRAM, and if this statement is

executed, the receiver does not allow this access to be executed and handles it as an error in accordance with the setting in application control information. The receiver operation in such a case shall be specified in operational rules. For example, the receiver may terminate the application and return to the display of the broadcast video in full-screen. Any transition to outside of the application boundary is prohibited. If Document a2 has a link to a document outside of the application boundary and if the user orders a transition to that document, the receiver handles it as an error. For example, the receiver may terminate the application and return to the display of the broadcast video in full-screen.

(4) When the user operates button b, the receiver fetches Document b1 from Domain-A, and makes a transition to it. This document is given all permissions just like the entry document. In the display of Document b1, the broadcast video continues to be displayed in a sub-screen, and Document b2, which is referred to in the iframe element, is displayed in a rectangular sub-screen inserted in the screen of Document b1. Since Document b2 is fetched from Domain-C, it is denied access to SI. If Document b2 has a statement that orders access to SI and if this statement is executed, the receiver does not allow this statement to be executed, and handles it as an error in accordance with the setting in application control information. The receiver operation in such a case shall be specified in operational rules. For example, the receiver may terminate the application and return to the display of the broadcast video in full-screen. Any transition to outside of the application boundary is prohibited. If Document b2 has a link to a document outside of the application boundary and if the user orders a transition to that document, the receiver handles it an error. For example, the receiver may terminate the application and return to the display of the broadcast video in full-screen.

(5) When the user operates button c, the receiver fetches Document c1 from Domain-A, and makes a transition to it. This document is given all permissions just like the entry document. In the display of Document c1, the broadcast video continues to be displayed in a sub-screen. If Document c1 has a link to a document outside of the application boundary and if the user orders a transition to that document, the receiver handles it as an error. For example, the receiver may terminate the application and return to the display of the broadcast video in full-screen.

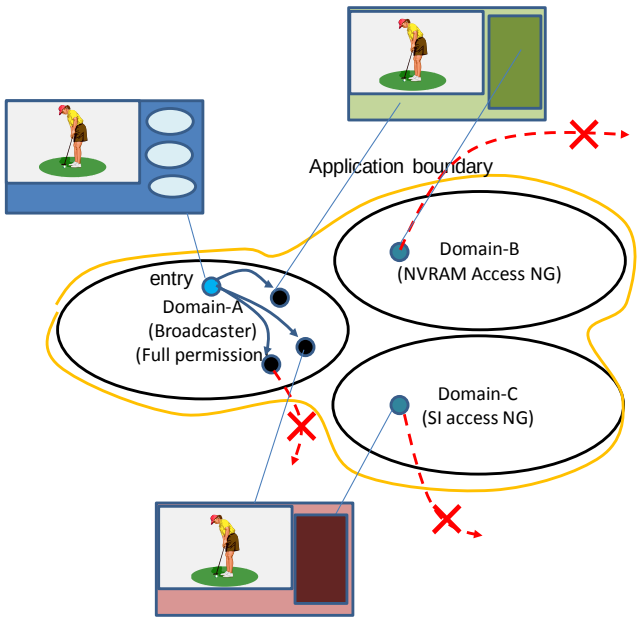


Figure C-14 Application using an iframe element in Case A

- **Scenario for Case B**

  Figure C-15 shows the concept of this case.

  (1) After the receiver has started to receive a broadcast service, it fetches and analyzes application control information, and holds the application boundary and access permission settings (application boundary and permission descriptor in AIT). Three Internet domains are set in the control information, as in Case A. Domain-A is a broadcaster domain, in which the application entry document is placed and in which access to all broadcast resources is permitted. Domain-B is a non-broadcaster domain, where access to NVRAM is not allowed. Domain-C is another non-broadcaster domain, where access to SI is not allowed. No access permission is given to the rest of the Internet domains, which are considered a fourth domain. Therefore, transitions to any domains other than the specified Domain-A, Domain-B and Domain-C are allowed, but HTML documents fetched from domains other than the above three domains are not allowed access to broadcast resources.

  (2) The receiver fetches the entry document of a broadcast-oriented managed application from Domain-A based on the application URL specified in the application control information, and executes it. Since access to all broadcast resources is allowed in Domain-A, the entry document is allowed to use various functions, such as referring to broadcast video, referring to SI, and accessing NVRAM. Therefore, the broadcast video can be displayed in a sub-screen, and buttons for a transition to Document a1, Document b1, or Document c1 can also

be displayed.

(3) When the user operates button a, the receiver fetches Document a1 from the same domain, Domain-A, and makes a transitions to the document. This document is also given all permissions just like the entry document. In the display of Document a1, the broadcast video continues to be displayed in a sub-screen, and Document a2, which is referred to in the iframe element, is displayed in a rectangular sub-screen inserted in the screen of Document a1. Since Document a2 is fetched from Domain-B, it is denied access to NVRAM. If Document a2 has a statement that orders access to NVRAM, and if this statement is executed, the receiver does not allow this access to be executed and handles it as an error in accordance with the setting in application control information. The receiver operation in such a case shall be specified in operational rules. For example, the receiver may terminate the application and return to the display of the broadcast video in full-screen.

(4) When the user operates button b, the receiver fetches Document b1 from Domain-A, and makes a transition to it. This document is given all permissions just like the entry document. In the display of Document b1, the broadcast video continues to be displayed in a sub-screen, and Document b2, which is referred to in the iframe element, is displayed in a rectangular sub-screen inserted in the screen of Document b1. Since Document b2 is fetched from Domain-C, it is denied access to SI. If Document b2 has a statement that orders access to SI and if this statement is executed, the receiver does not allow this statement to be executed, and handles it as an error in accordance with the setting in application control information. The receiver operation in such a case shall be specified in operational rules. For example, the receiver may terminate the application and return to the display of the broadcast video in full-screen.

(5) When the user operates button c, the receiver fetches Document c1 from Domain-A, and makes a transition to it. This document is given all permissions just like the entry document. In the display of Document c1, the broadcast video continues to be displayed in a sub-screen, and Document c2x, which is referred to in the iframe element, is displayed in a rectangular sub-screen inserted in the screen of Document c1. In the initial state, Document c2x within the iframe is Document c21, which was fetched from Domain-D. From there, a transition can be made by a user operation to Document c22 in Domain-E, Document c23 in Domain-F, or any document in any domain by following links. Since documents c2x are in domains outside Domain-A, Domain-B or Domain-C, they are denied access to broadcast resources in accordance with the application control information. If Document c2x has a statement that orders access to any broadcast resources and if this statement is executed, the receiver does not allow this access to be executed, and handles it as an error in accordance with the setting in application control information. The receiver operation in such a case shall be specified in operational rules. For example, the receiver may terminate the application and

return to the display of the broadcast video in full-screen.

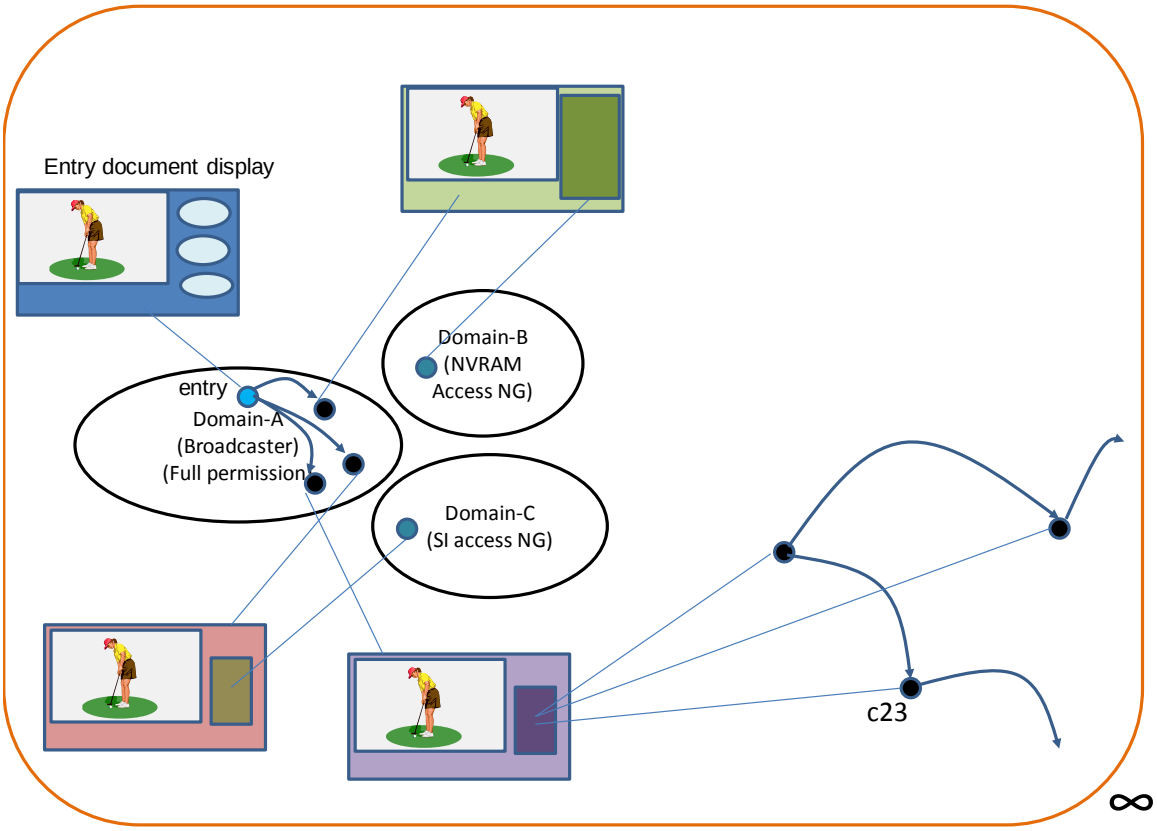


Figure C-15 Application using an iframe element in Case B

Appendix D Service Examples

In order to aid the reader in getting a general idea about the services addressed in this system, this appendix presents examples of screen transitions and service configurations.

D.1 Broadcast-oriented managed application

D.1.1 Program recommendation

The transitions of the application screens depend on the design of each service. Figure D-1 shows screen transitions using the application for a “program recommendation” service as an example.

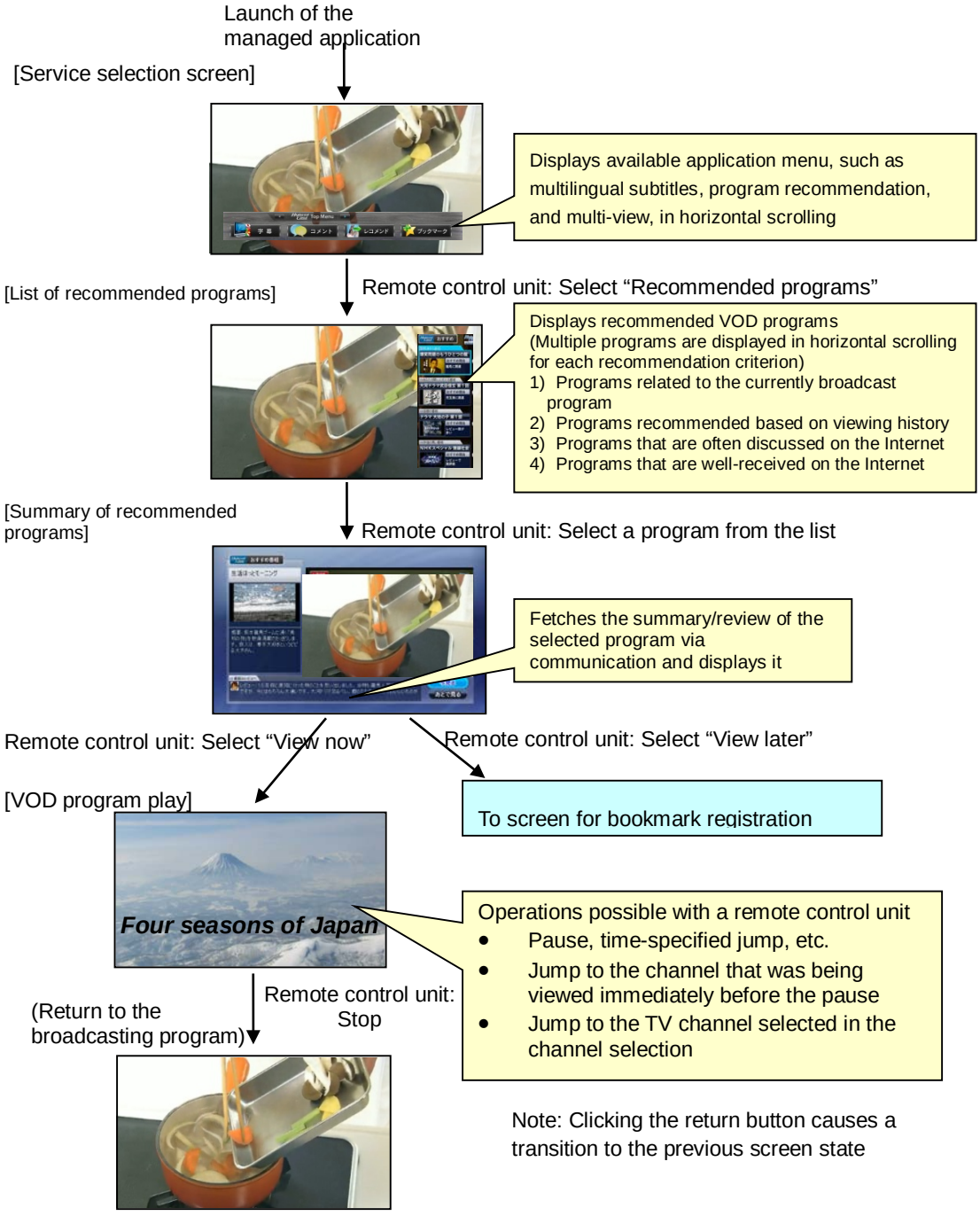


Figure D-1 Example of a program recommendation application

D.1.2 Companion device-coordinated service associated with commercials

Figure D-2 shows an example of presenting an application that works with a commercial that is being displayed using the coordinated operation function between the receiver and an external device. Application screen transitions are dependent on the design of each service. In this service, in

synchronization with some timing in a broadcast commercial, the related commercial application is presented on the TV screen. The receiver communicates with the external device with the same timing so that a related commercial application is also displayed on the external device. In this service example, a companion device-coordinated application that works with the TV and the external device is presented on the screen, and data is exchanged between them. Companion applications run on external devices and have the function to work with the receiver. Broadcaster applications are assumed to have an equivalent function.

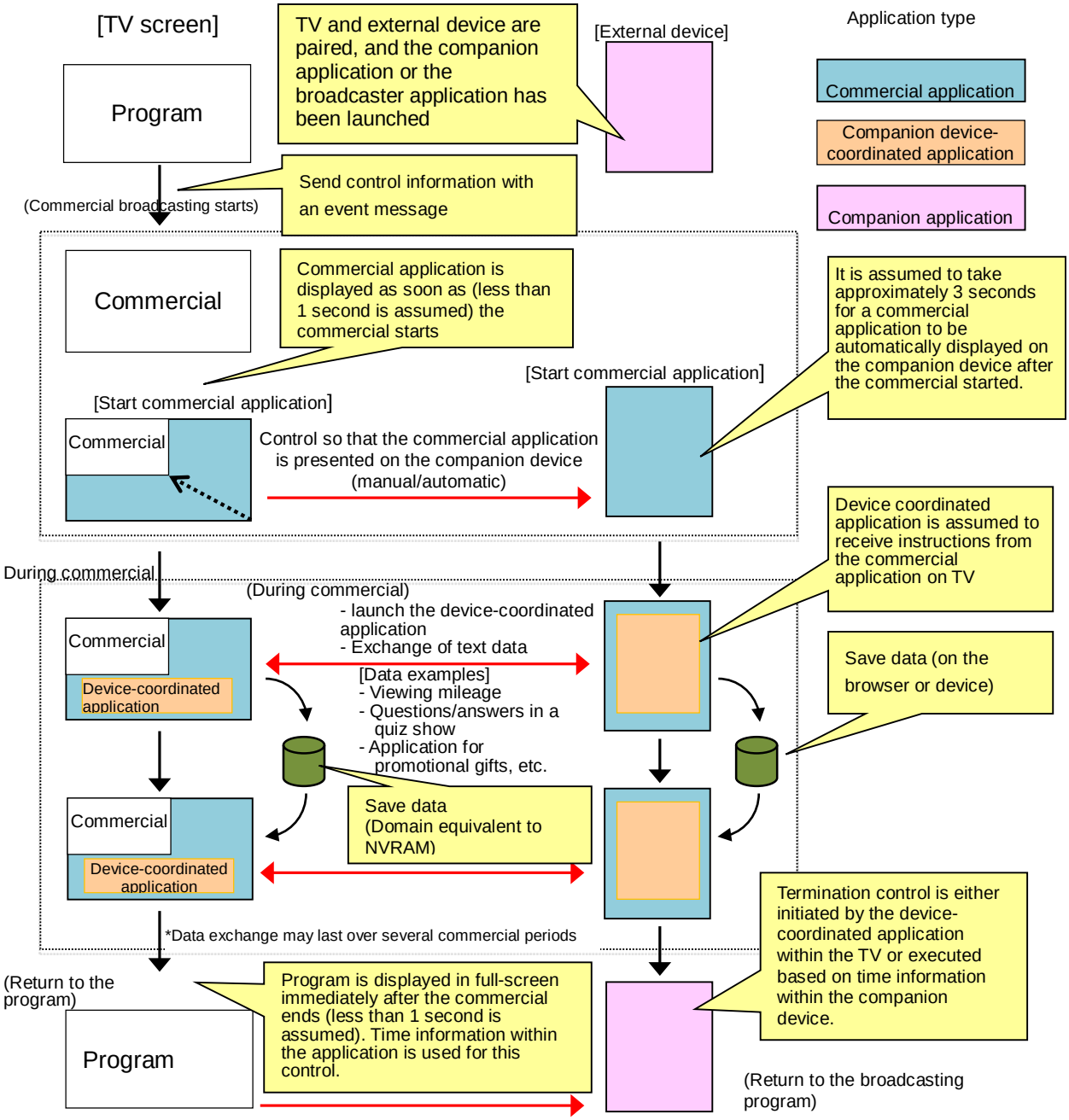


Figure D-2 Example of companion device-coordinated services integrated with commercials

D.1.3 Sharing messages with data broadcasting

An application service associated with a specific program is assumed here. The viewer of a TV that supports the integrated broadcast-broadband function launches the associated application and views it, and the viewer of a TV that supports BML views data broadcasting associated with the program. In this state, the broadcaster sends, or prepares to send, the same message to both these types of TV so that the TVs will behave the same way. An example is given in Fig. D-3.

The broadcaster sends an event message at the time when the scene is switched. When the TVs receive this message, the application or data broadcasting on the TVs switches the display content at the same time. Similarly, when the broadcaster re-writes the files uploaded on the server, the application and data broadcasting change the displayed character string and images at the same time.

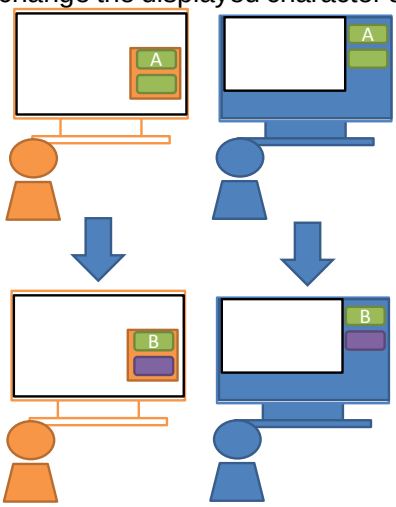


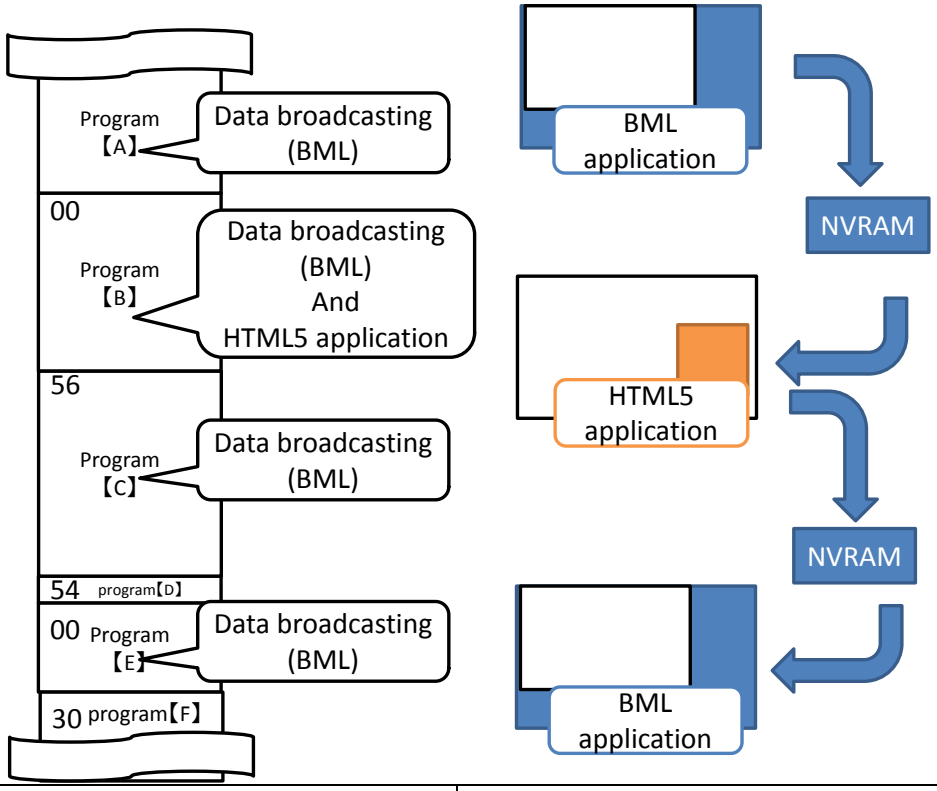
Figure D-3 Example of TVs sharing messages in the form of HTML and BML content to enable the TVs to behave in the same way
(Left: HTML5 application. Right: data broadcasting associated with the program)

D.1.4 Integrated service that handles both BML content and HTML applications

Both BML content and an HTML application access a domain that stores information unique to each type of TV. This example concerns a program plan that involves multiple programs, for example, enabling the viewer to accumulate loyalty points in proportion to their viewing hours across different programs during a certain campaign period. Figure D-4 shows an example of program scheduling and the concept of this service.

We assume a case where a program with which only BML data broadcasting is associated is mixed with a program with which both an HTML application and BML data broadcasting are associated. The associated data broadcasting writes earned points for each device into NVRAM. Then, an application associated with the next program reads data from NVRAM and writes earned points into it. As a result,

information unique to each device is shared by BML content and HTML applications via NVRAM.



日本語	English
ニュース good イブニング[B]パンダの赤ちゃん	News Good Evening [B] Panda's baby
ヌーメロンでショウ[B][H] 今日は豪華ゲストが盛りだくさん	Numeron-de-Show [B][H] There are many gorgeous guests today
60 億人の逆質問!?[B] SP ゲストの弱点発覚! 海外で活躍するモデルの夏休み	Reverse questions from 6 billion people[B] Weak points of special guests discovered! Summer vacation of fashion models who are successful overseas
ニュース[B]	News [B]
おしゃべり○○[B]	Talk Show xx [B]
BML のみ連動	Only BML content is associated
BML・HTML 連動	Both BML content and HTML application are associated
BML コンテンツ	BML content
HTML5 アプリ	HTML5 application

Figure D-4 Example of a program-crossing plan over BML and HTML content

D.1.5 Paid service

This example concerns the presentation of program/channel information and an invitation to non-subscribers to subscribe to the service. The application needs to be designed with care with regard to reading and sending personal information, such as subscriber ID. Figure D-5 shows the overall configuration of the service.

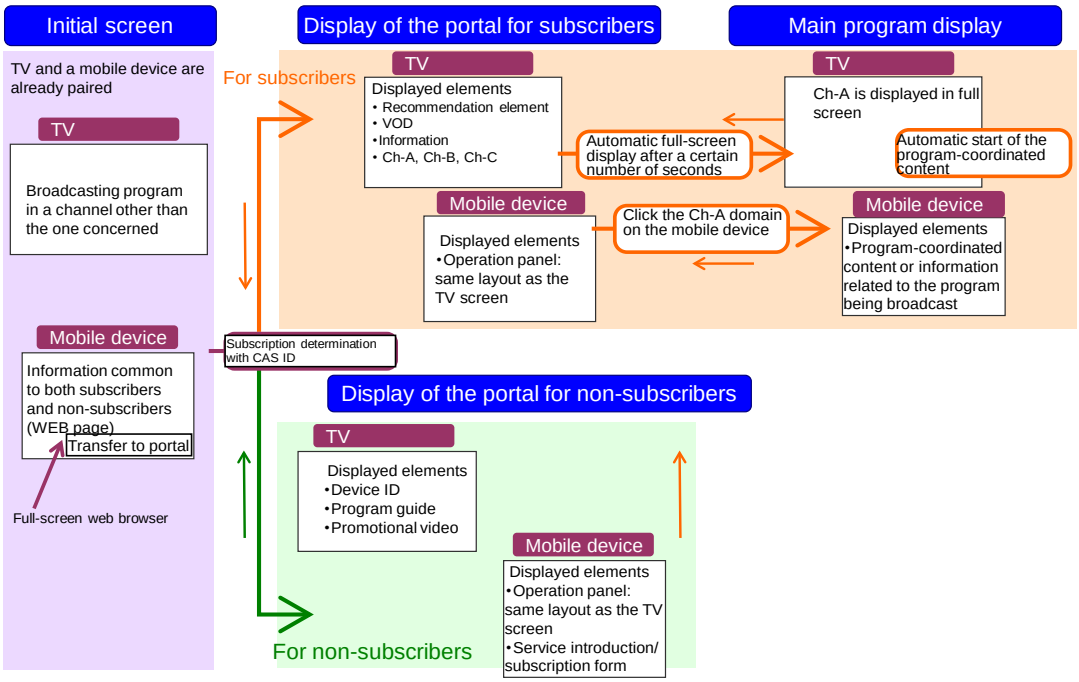


Figure D-5 Concept of the paid channel portal service

Figure D-6 shows examples of the information screens for the subscriber. This service presents information related to the broadcast content on the mobile device.

- While the main program is being broadcast on TV, the mobile device is made to display an HTML document that presents information about the program or information about programs that are currently being broadcast on other channels.
- The mobile device is made to display the relevant HTML document in a size appropriate for the position in the program frame. While the program advertisement is being broadcast, the TV program screen is reduced and a guide to the next program frame is presented in the L-shaped window.
- This is a regular arrangement that applies to all broadcasting frames.
- Selection of another channel is also possible.

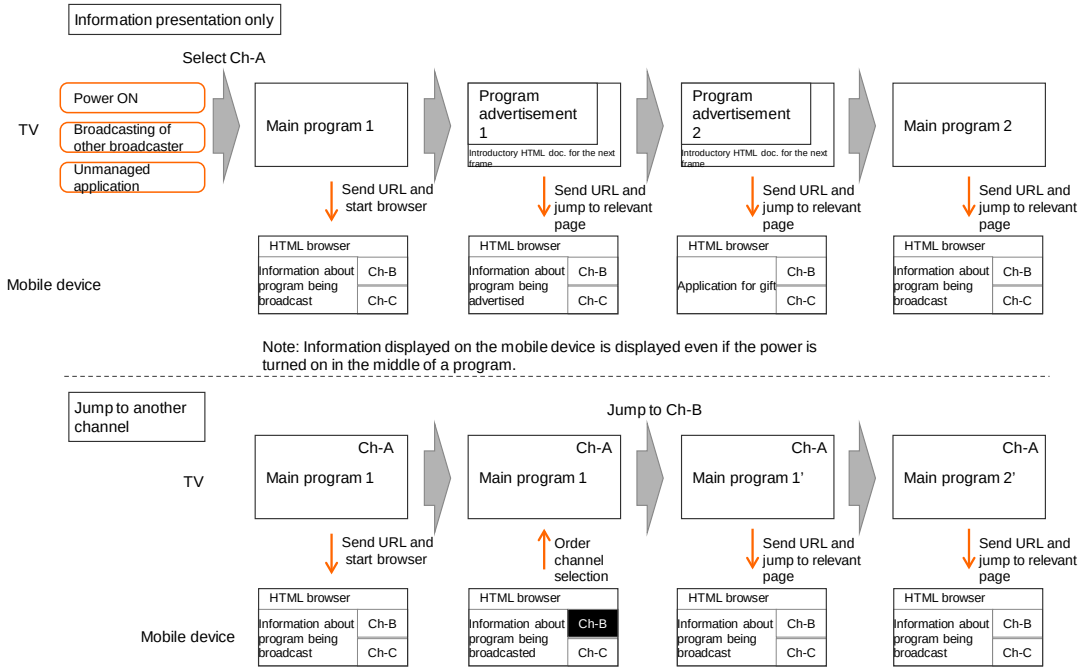
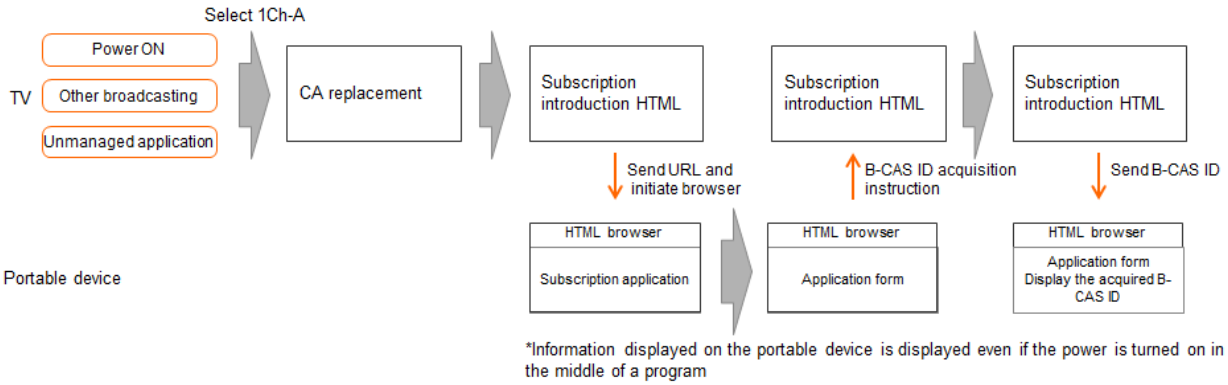


Figure D-6 Examples of the information screens for the subscriber

Figure D-7 shows examples of the information screens for a non-subscriber. For a non-subscriber, information about subscription to the paid service is presented on the mobile device.

- Regardless of the program being broadcast, an HTML document presenting information about subscription to the paid broadcasting service is displayed on the mobile device.
- The HTML document is displayed in full-screen instead of the TV program.
- The HTML document for subscription application fetches and displays the TV device ID.



日本語	English
他局の放送	Broadcasting of other broadcaster

加入案内 HTML	Subscription information HTML doc.
URL を送りブラウザを起動	Send URL and start browser
端末 ID 取得指示	Order to fetch device ID
端末 ID を送る	Send device ID
取得した端末 ID を表示	Display fetched device ID
※携帯端末に表示させる情報は、番組の途中で電源を入れても表示される。	Note: Information displayed on the mobile device is displayed even if the power is turned on in the middle of a program.

Figure D-7 Examples of the information screens for a non-subscriber

D.2 Non-broadcast-oriented managed applications

An example from the launch of a non-broadcast-oriented managed application to its termination is shown in Fig. D-8.

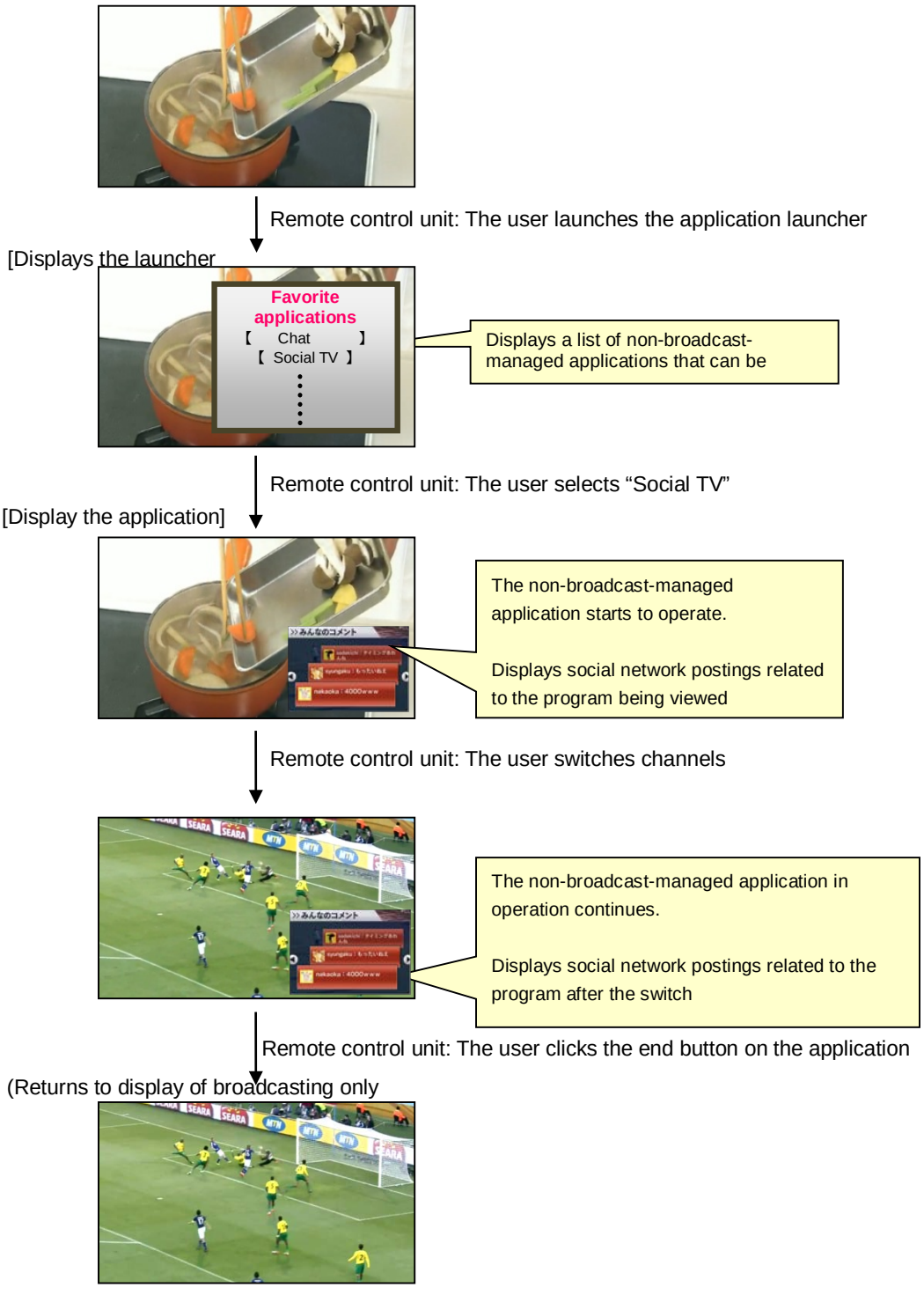


Figure D-8 Example of a non-broadcast-oriented managed application

D.3 Launch and termination of an integrated broadcast-broadband service

Figure D-9 shows an example of transitions from the launch to termination of an integrated broadcast-broadband service in this system.

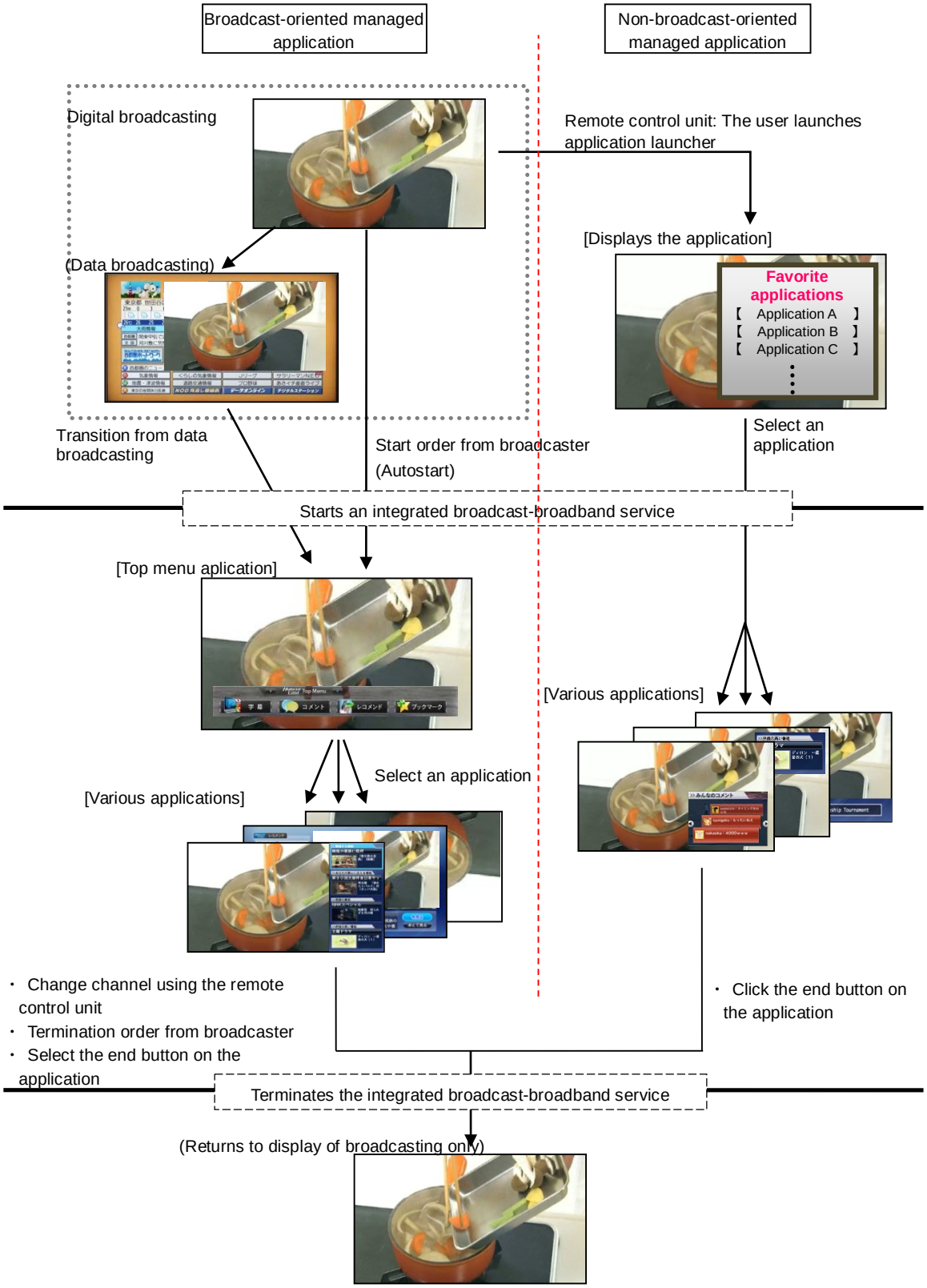


Figure D-9 Example of the launch and termination of an integrated broadcast-broadband service

There are three ways to launch an integrated broadcast-broadband service:

1) If there is an application with automatic start specified in the application control information, this application is displayed automatically. (Broadcast-oriented managed application)
2) A transition to an integrated broadcast-broadband service is made when the function that causes a transition from BML data broadcasting to an integrated broadcast-broadband application in this system is invoked. (Broadcast-oriented managed application)
3) The user calls the application launcher of the receiver using a remote control unit, and selects an application. (Non-broadcast-oriented managed application)

An integrated broadcast-broadband service is terminated when the application concerned is terminated in one of the following ways:

1) The application is terminated when the channel is switched. (Broadcast-oriented managed application)
2) The application is terminated at the instruction of application control information transmitted by the broadcaster. (Broadcast-oriented managed application)
3) If the application has a termination button, the user clicks that button to terminate the application. (Broadcast-oriented managed application and non-broadcast-oriented managed application)

D.4 Display of subtitles

There are the following issues in handling broadcast subtitles when an application is being presented in this system:

- Target subtitle data

  Considering operation cost and support of legacy receivers, text information for subtitle data for the 12-segment video defined in ARIB STD-B24 is used for the presentation of subtitles.

- Visibility of subtitles and applications

  It is necessary to ensure the visibility of subtitles even when the video is reduced in size by an application.

  However, it is recommended that the burden on the application be minimized. One way to ensure this is to display subtitles outside of the HTML browser when an application is being presented, regardless of whether or not the application content and video are displayed at a reduced size.

- Method of presenting broadcast subtitles

  The position and size of subtitles shall be controlled by receiver functions. Guidelines as to the display position and size (DPI) of subtitles when an application is presented may be specified in operational rules. (Subtitles may be displayed on an area at the very bottom of the TV screen as shown in Figure D-10.) In addition, it is recommended that the browser size can be fetched when subtitles are displayed by an application.

  One way to determine whether an application should be displayed is to use information as to whether a Fullscreen API call is used. However, it is necessary to study the matter in more detail. It is also necessary to study the method of enabling the application to specify the subtitle display position, including whether such a method is required.

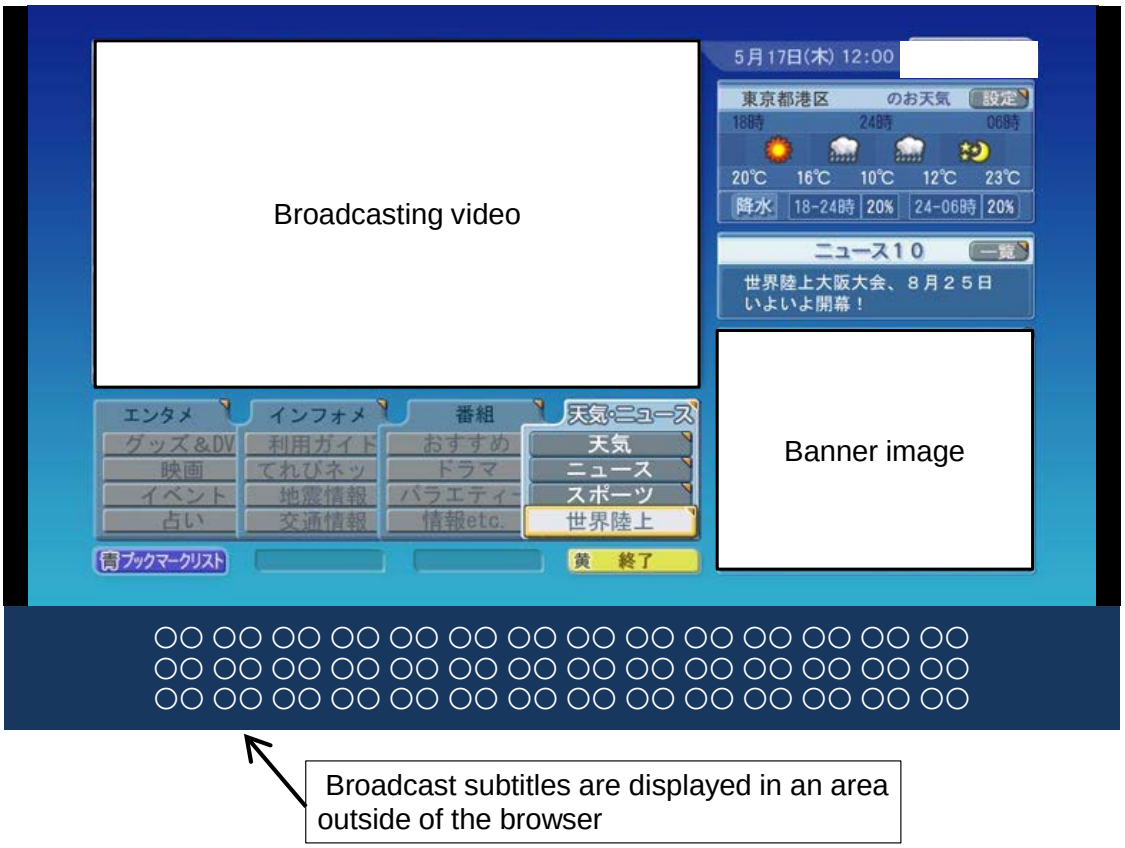


Figure D-10 Example of broadcast subtitle display

D.5 Control of the presentation of EWS broadcasting

If an EWS (emergency warning system) broadcast is received, the subtitles and broadcast program that convey this information are given priority, and the presentation of the ongoing application is temporarily disabled. Figure D-11 shows an example of such control. Although the application is overlaid on the broadcasting screen in Fig. D-11, regardless of whether the application is overlaid or not, the display of the application is temporarily removed, and the broadcast program is displayed in full-screen.

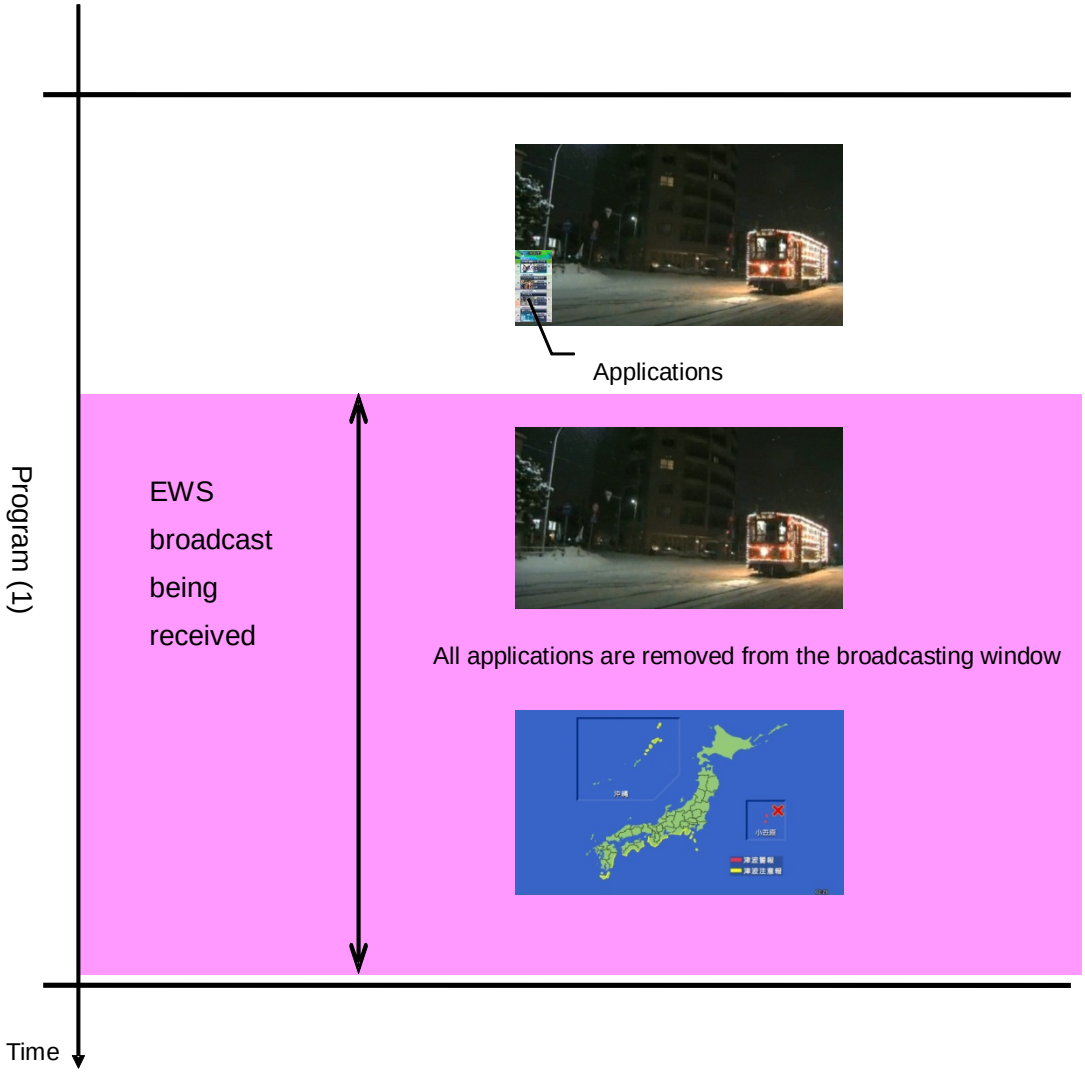


Figure D-11 Example of presentation control at the time of EWS reception

D.6 Coordinated operation with a mobile device

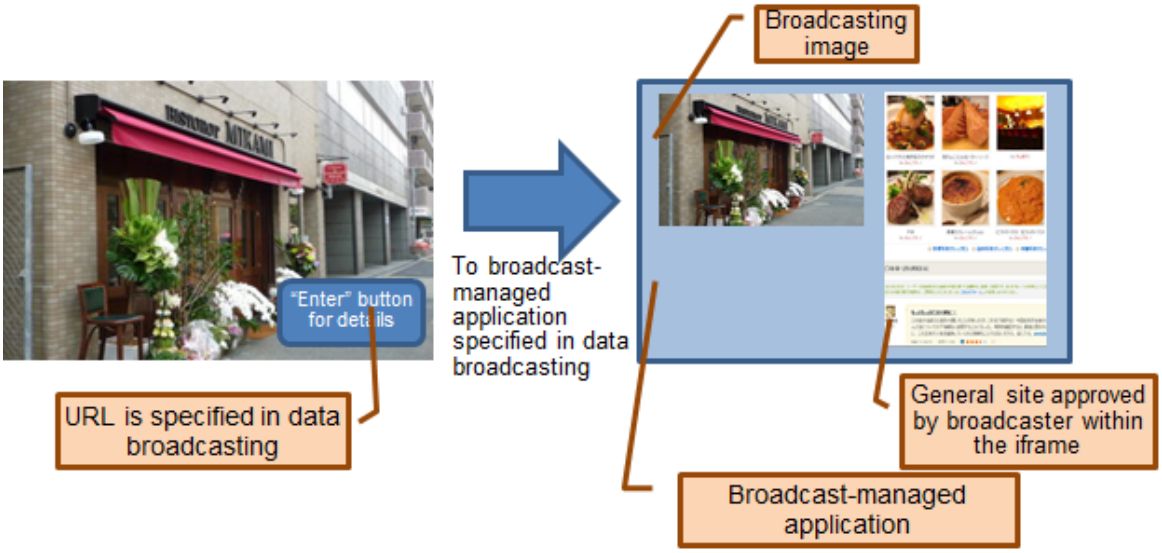
Figure D-12 shows an example of a service for information search through coordinated operation between the receiver and a mobile device. The mobile device is used as a second screen. The receiver forwards the program summary and key words related to the program scenes received from the broadcast program to the mobile device, and the mobile device displays them. At the same time, the mobile device performs web search based on the forwarded key words, and displays the search result.

Figure D-12 Example of information search through coordinated operation with a mobile device

D.7 Synthesized presentation of a broadcast program and an application that includes access to sites operated by a provider other than the broadcaster

Examples of synthesized presentation of a broadcast program and an application that includes access to sites operated by a provider other than the broadcaster are shown in Figs. D-13 and D-14. In Fig. D-13, data broadcasting is used to display information encouraging the user to launch a certain application, and the user launches the application in accordance with this information. (The information may be displayed on a button that causes a transition to the application.) When the application is launched, the broadcast program is displayed at a reduced size. At the same time, a

website approved by the broadcaster is displayed using an iframe element. The URL of the website displayed in the iframe element is updated as the program progresses.



Current expression	Revised expression
“Enter” button for details	Click here for details
To broadcast-managed application specified in data broadcasting	To a broadcast-oriented managed application specified in data broadcasting
Broadcasting image	Broadcast video
General site approved by broadcaster within the iframe	A broadcaster-approved general site displayed within the iframe
Broadcast-managed application	Broadcast-oriented managed application

Figure D-13 Example of synthesized presentation of a broadcast program and an application

Figure D-14 shows an example in which the website URL in the example of Figure D-13 is updated as the program progresses by event messages multiplexed with the broadcast signals. After a transition to a broadcast-oriented managed application, the website content displayed within the iframe can be updated as the program progresses.

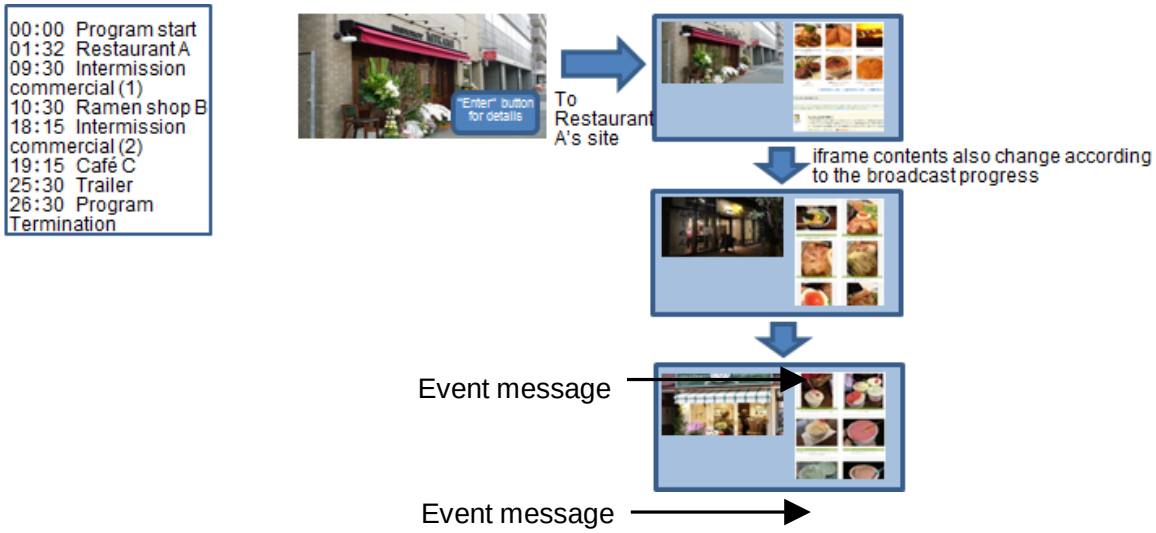


Figure D-14 Example in which presentation contents change in accordance with the progression of the program

Current expression	Revised expression
iframe contents also change according to the broadcast progress	As the program progresses, the iframe content is updated

D.8 Scenario for companion device collaboration service in System model B for companion device collaboration

Assumed scenario

It is assumed that the user installs and launches a collaboration application (collaboration app) in his/her programmable mobile terminal, typically a smartphone or a tablet. This app provides companion device collaboration functions, such as enabling the mobile terminal to discover the receiver and to exchange data with the receiver. For example, a broadcaster or a conglomerate of broadcasters may offer a native app that provides a second screen service. This scenario does not assume that webpages are displayed on the receiver's screen using HTML5. The user launches a collaboration app while he/she is viewing a broadcast service. The collaboration app communicates with a function built into the receiver to receive data contained in the broadcast signal that is received by the receiver. While the receiver is receiving a broadcast service and while a communication is established between this built-in function and the collaboration application, the function constantly sends data to the collaboration app. The collaboration app can change the received channel or turn the receiver's power on or off using the API for access to the receiver function. Details of this API are published from time to time.

Figures D-15, D-16 and D-17 show, respectively, transitions of the screen display, the software configurations of the receiver and the mobile terminal, and a typical operation sequence in a companion device collaboration scenario.

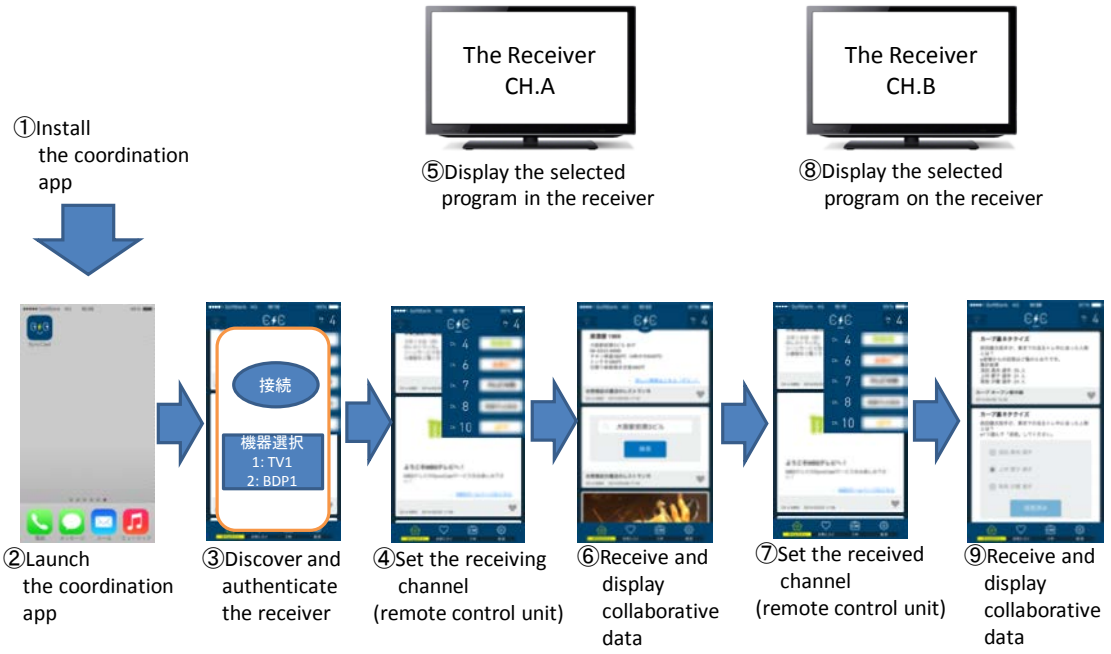
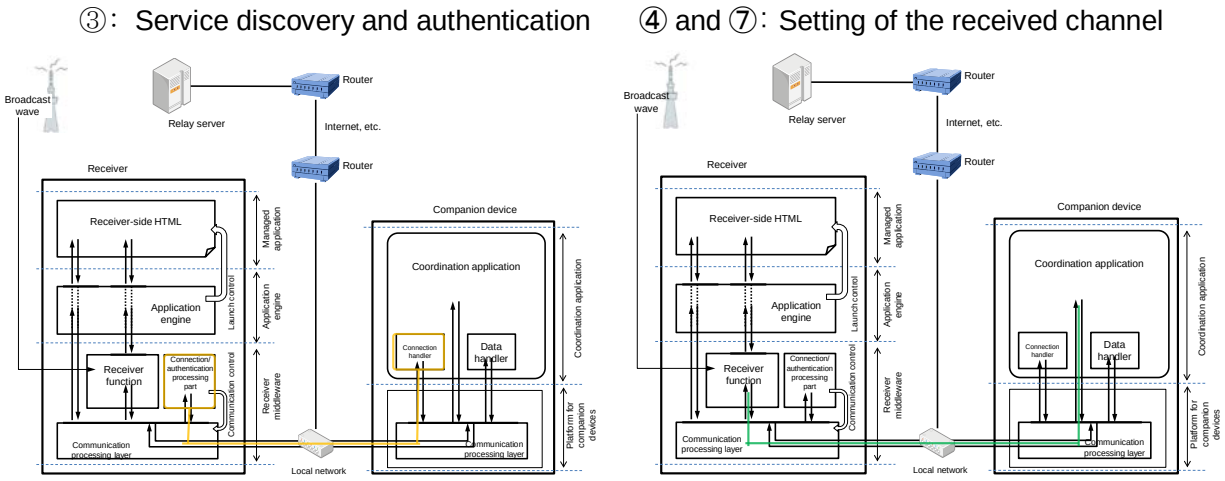


Fig. D-15 Screen transitions in companion device collaboration



⑥ and ⑨: Receive and display collaboration data

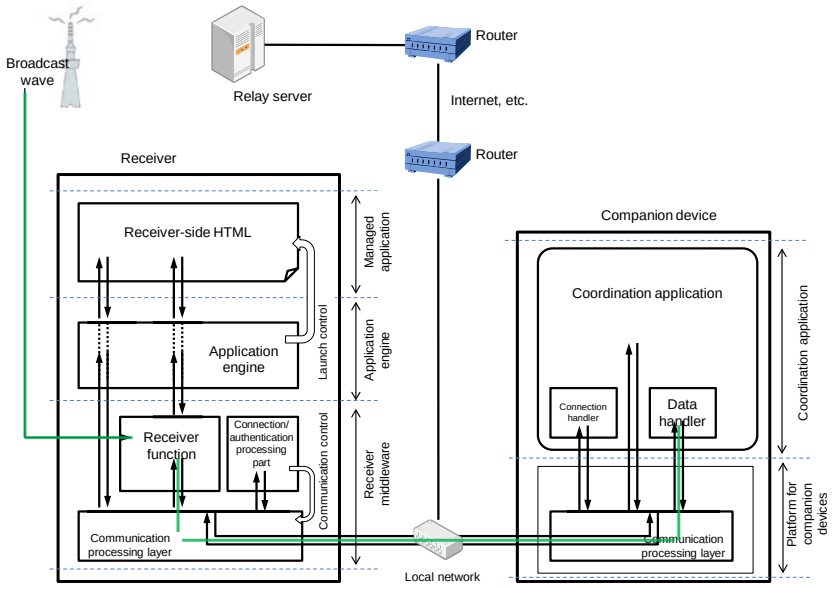


Fig.D-16 Examples of software configurations of the receiver and the mobile terminal

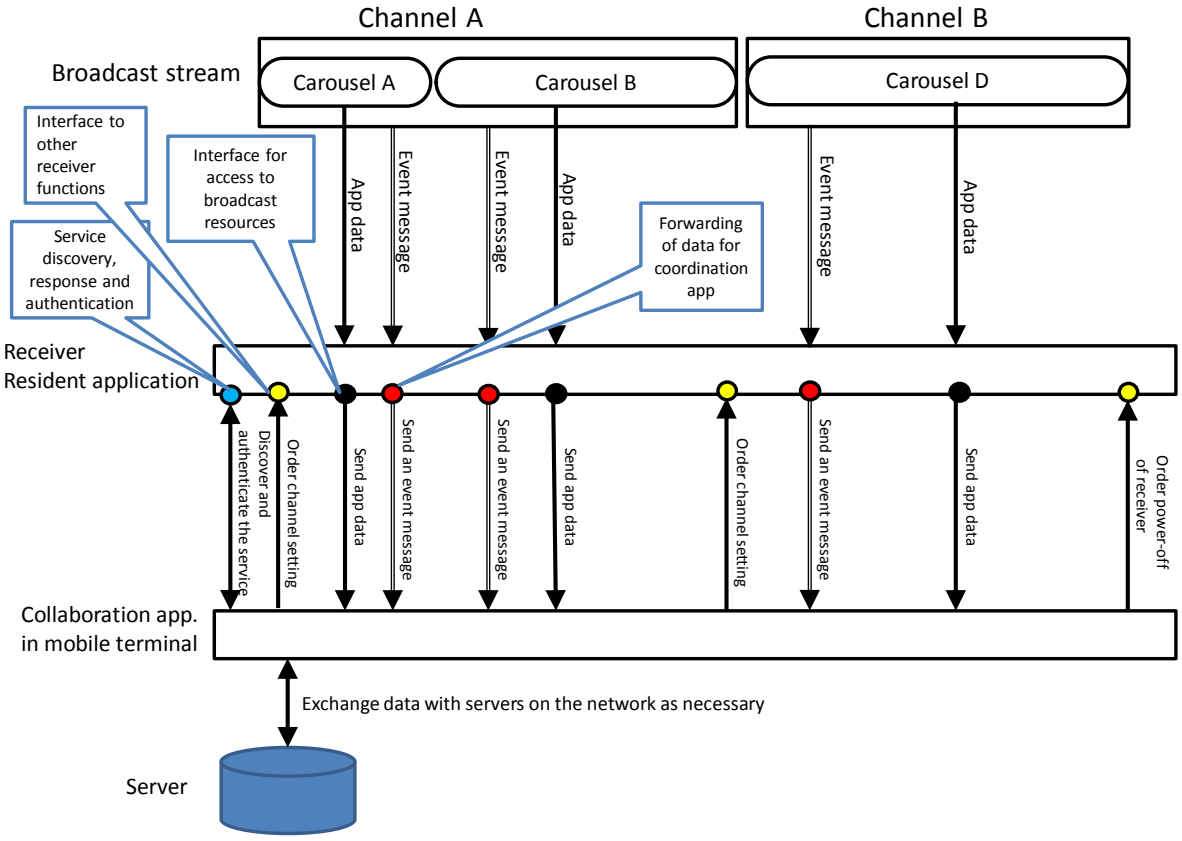


Fig.D-17 Example of operation sequence in companion device collaboration

Operation sequence

1. The viewer installs the collaboration app in his/her mobile terminal (companion device).
2. The viewer selects and launches the collaboration app. The collaboration app discovers the service, selects the receiver it intends to connect to, and sends a connection request to it. The receiver that has received this connection request authenticates the request sender. If the receiver confirms the authenticity of both the companion device and the collaboration app, it accepts the request, and the connection is established.
3. The collaboration app accesses the “interface to other receiver functions” and orders channel selection. The receiver selects the broadcast service, starts playing its audio/video data, and receives data for the collaboration app and an instruction to transfer this data to the collaboration app.
4. When the receiver has received the app data contained in the broadcast signal, it forwards the data to the collaboration app via the “broadcast resource access interface.” The collaboration app processes the app data and takes the appropriate action.
5. When the receiver receives an event message contained in the broadcast signal, it forwards the message to the collaboration app via the “forwarding of data to the collaboration app.” The collaboration app processes the event message it has received and takes the appropriate action.
6. If the collaboration app needs to change channels, ③ to ⑤ are repeated.
7. The collaboration app accesses the “interface to other receiver functions” and orders the receiver’s power-off.

D.9 Scenario for companion device collaboration service in System model D for companion device collaboration

Assumed scenario

If collaboration apps that are independent of receiver models are available, this will be convenient for the user, promote the use of such apps, and increase the number of those who want to develop such apps.

To realize collaboration apps that are independent of receiver models, it is necessary to specify an API through which collaboration apps can invoke collaboration commands that can be used universally with any general receivers. However, to implement a standardized API in receivers, it may be necessary to change a receiver-specific design or receiver implementation, and it may be difficult to support receiver vendor-specific commands satisfactorily.

A possible solution to these problems is to introduce a mechanism that converts messages through a standardized API into receiver-specific command types. This mechanism allows receiver vendors to implement receiver models in their own way while still supporting a standardized API.

The following is a possible operation sequence for a companion device collaboration service that incorporates the above mechanism. In this example, a command conversion table is placed in a relay server.

Operation sequence

1. The viewer installs the collaboration app in his/her mobile terminal (companion device).
2. The viewer selects and launches the collaboration app. The collaboration app connects to a relay server, and authenticates it. It then selects the receiver it intends to connect to, and sends a pairing request to it. The relay server that has received the pairing request sets up pairing between the companion device and the receiver. If the pairing is successful, a secure connection is established between the companion device and the receiver.
3. The collaboration app sends to the relay server a standardized command (e.g., an instruction to select a channel) that is independent of specific receiver models.
4. The relay server has a command conversion table for each receiver model. This command conversion table is used to manage the correspondence between standardized commands and receiver model-dependent commands. When the relay server receives a command from the collaboration app, it converts the command to the appropriate receiver model-dependent command by referring to the command conversion table, and sends the latter command to the receiver with which the companion device is paired.
5. When the receiver receives a command from the relay server, it accesses the receiver function to execute the received command, and sends the execution result to the relay server.
6. When the relay server receives the result from the receiver, it converts the result to a standardized command execution result by referring to the command conversion table, and sends

the latter result to the collaboration app.

7. When the collaboration app receives the command execution result, it takes action appropriate for the result.

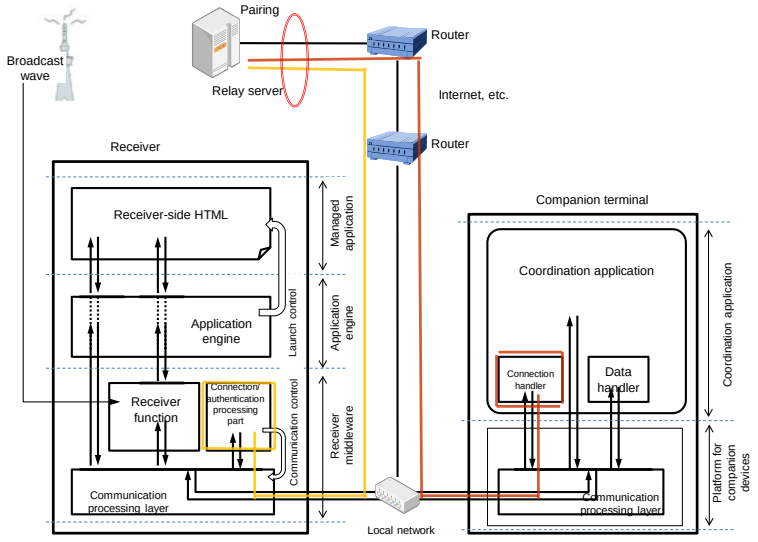


Fig.D-18 Companion device collaboration sequence example: authentication and pairing

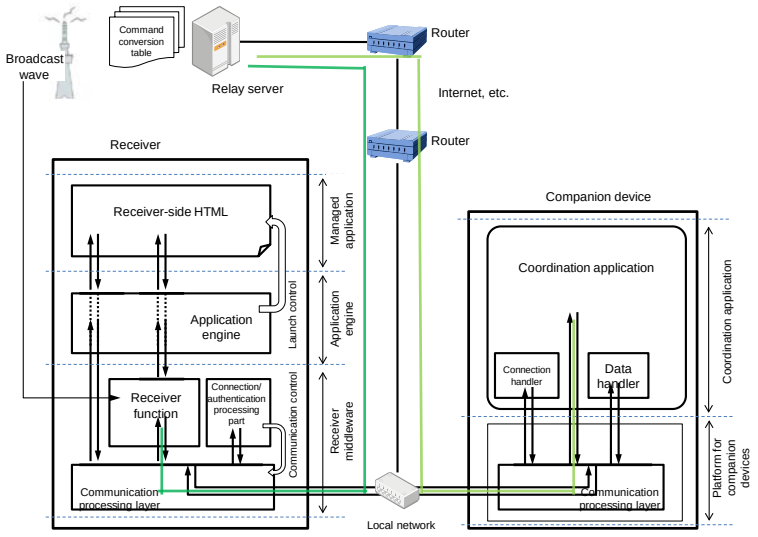


Fig.D-19 Companion device collaboration sequence example: command conversion

Appendix E

Appendix E is intentionally left blank.

Appendix F Control of a Non-broadcast-oriented Managed Application through the Broadcast Signal

Access by a non-broadcast-oriented managed application to broadcast resources is in principle controlled based on application control information (t-AIT) as specified in Chapter 14. Minute-by-minute access by a broadcast-oriented managed application to broadcast resources in a particular service can be controlled based on another item of application control information contained in the broadcast signal. While a non-broadcast-oriented managed application is running, the receiver controls access by a specific application to broadcast resources at a specific time for a specific broadcast service by checking whether the above two items of application control information match. To make it possible to control non-broadcast-oriented managed applications from the broadcast signal, it is assumed that the following draft specification will be added to ARIB STD-B24 Volume 4.

F.1 Transmission of non-broadcast-oriented managed application control information in the broadcast signal

The control information of a non-broadcast-oriented managed application shall be transmitted in an application information table (AIT) as specified in ARIB STD-B24 Volume 4 5.1. An external application control descriptor as specified in F.2, which is information used to control a non-broadcast-oriented managed application, shall be placed in the common descriptor area of the AIT concerned. Two modes of AIT can be assumed: a mode in which an AIT serves as control information for both a broadcast-oriented managed application and a non-broadcast-oriented managed application, including a description of an application loop, and a mode in which an AIT is dedicated to the control of a non-broadcast-oriented managed application in which no application loop exists. Access permission for a non-broadcast-oriented managed application in cases where the AIT (the AIT including an external application control descriptor) specified in this appendix is not transmitted shall be specified in the operational rules.

When it is necessary to distinguish the AIT that is transmitted over the broadcast signal and defined in this appendix from the XML-format AIT specified in Chapter 14, the former AIT is called “b-AIT” in this document.

F.2 External application control descriptor

The data structure of an external application control descriptor is shown in TableF-1.

Table F-1 External application control descriptor

Data structure	Bit count	Bit string notation

external_application_control_descriptor(){		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
specific_scope_flag	**1**	**bslbf**
reserved_for_futureuse	**7**	**bslbf**
if(specific_scope_flag==1){		
application_class_scope	**16**	**bslbf**
application_count	**8**	**uimsbf**
for(i=0;i<application_count;i++){		
application_identifier()	**48**	**uimsbf**
}		
}		
permission_bitmap_count	**8**	**uimsbf**
for(i=0;i<permission_bitmap_count;i++){		
permission_bitmap	**16**	**bslbf**
}		
overlay_admission_polarity	**1**	**bslbf**
reserved_for_future_use	**3**	**bslbf**
overlay_controlled_area_count	**4**	**uimsbf**
for(i=0;i<overlay_controlled_area_count;i++){		
overlay_controlled_area_tag	**8**	**uimsbf**
upper_left_horizontal	**16**	**uimsbf**
upper_left_vertical	**16**	**uimsbf**
horizontal_size	**16**	**uimsbf**
vertical_size	**16**	**uimsbf**
}		
blocked_application_count	**8**	**uimsbf**
for(i=0;i< blocked_application _count;i++){		
application_identifier()	**48**	**uimsbf**
}		
}		

This descriptor indicates default access permission or access permission given to a specific group of applications (hereafter called the “scope” of the permission), and indicates the video area where overlaid display is either permitted or prohibited.

- Semantics
  - **specific_scope_flag**. This is a flag for a specific group of applications. The value “0” for this flag means the default setting, in which the access permission setting applies to all

applications. If this flag is “1,” the access permission setting is applied to a specified group of applications. If a number of descriptors of this type are placed, including a default setting and settings for specific groups of applications, settings for applications not included in the identified specific groups shall be the default setting. By placing a number of descriptors of this type, it is possible to set access permission for each of multiple groups of applications. In such a case, the setting that appears earliest is given priority. Therefore, it is necessary to take care to place the settings for specific groups of applications in order of priority and to place the default setting at the end.

- **application_class_scope**. Specifies the application classes covered by the access permission setting. Application classes that are covered by the access permission setting are indicated using the following bitmap.

Bit	Application class
8 - 15	Reserved for use in the future
7	Broadcast video inclusion class
6	widget class
5	No-display class
4	Packaged class
3	Host class
2	certificationClass is not specified
1	Method 2 authentication class
0	Method 3 authentication class

If a number of bits are set to “1,” the scope of permission for application classes is determined by applying an “OR” operation to the scope of each class.

- **application_count**: number of applications. This shows the number of applications that are covered by the access permission setting.
- **application_identifier()**. This identifies the applications covered by the access permission setting. It is written using application identifiers as specified in 14.2. An application shall be covered by the access permission setting when its identifier matches this application identifier or it is within the scope of the specified application class.
- **permission_bitmap_count**: number of access permission bitmaps.
- **permission_bitmap**: access permission bitmap. This shows whether access to each broadcast resource is permitted or not in a bitmap of 16 bits, each bit being associated with a specific function. The first 3 bits indicate switching of bitmaps. The assignment of a functional bitmap shall be specified in the operational rules. If it is determined that an application does not have access permission, the receiver shall blocks its access to broadcast resources.

- ✧ **overlay_admission_polarity**: video overlay admission polarity. This indicates whether overlay is permitted in the area specified by the following field. The value “1” means that overlay is permitted, and “0” means that it is prohibited.
- ✧ **overlay_controlled_area_count**: The number of video overlay controlled areas.
- ✧ **overlay_controlled_area_tag**: video overlay area tag. This is the identification number of a rectangular video overlay area as specified below.
- ✧ **upper_left_horizontal**: horizontal coordinate of the upper left corner of the video overlay area. This is indicated in terms of pixel count.
- ✧ **upper_left_vertical**: vertical coordinate of the upper left corner of the video overlay area. This is indicated in terms of pixel count.
- ✧ **horizontal_size**: width of the video overlay area. This is indicated in terms of pixel count.
- ✧ **vertical_size**: height of the video overlay area. This is indicated in terms of pixel count.
- ✧ **blocked_application _count**: number of blocked applications. This indicates the number of applications that are in a blacklist specified below and are thus unconditionally denied access to broadcast resources.
- ✧ **application_identifier()**: application identifier. This indicates an application that is unconditionally denied access to broadcast resources. This is entered using an application identifier as specified in 14.2.

F.3 Example of a scenario in which a non-broadcast-oriented managed application is controlled through the broadcast signal

- ■ Scenario 1: access control without an area specified

① The user selects and launches a non-broadcast-oriented managed application using a launcher. In this example, the application concerned is a widget class application.

② The terminal fetches the t-AIT, and verifies the application authentication. If the verification is successful, it fetches the application and launches it.

③ The terminal also fetches the b-AIT being sent in the broadcast service selected, applies an “AND” operation to the access permission bitmap specified in the external application control descriptor and the access permission bitmap specified in the broadcastPermission element of the t-AIT, and uses the result as access control information while the application is running. In this example, overlay display (without an area specified) is permitted based on the b-AIT and the t-AIT of Service1.

④ When the terminal launches the application, it determines whether access to broadcast resources is permitted based on the access control in ③, and reflects the result in the execution of the application. In this example, overlay display on the video of Service1 is permitted. Therefore, the application is displayed on the display area of the broadcast video.

⑤ The user switches from Service1 to Service2.

⑥ The terminal performs channel selection, and begins to receive and play the broadcast stream of the newly selected service. At the same time, the terminal fetches the b-AIT associated with the newly selected service. As in ③, it applies an “AND” operation to the access permission bitmap of the t-AIT and the access permission bitmap of the b-AIT, and uses the result as new access control information. In this example, overlay is not permitted in Service 2. As a result, the control information indicates that overlay display is not permitted.

⑦ Using the above control information, access by the application to broadcast resources is reevaluated, and the result is reflected in access control. In this example, overlay display is not permitted. As a result, the application is displayed outside the display area of the broadcast video.

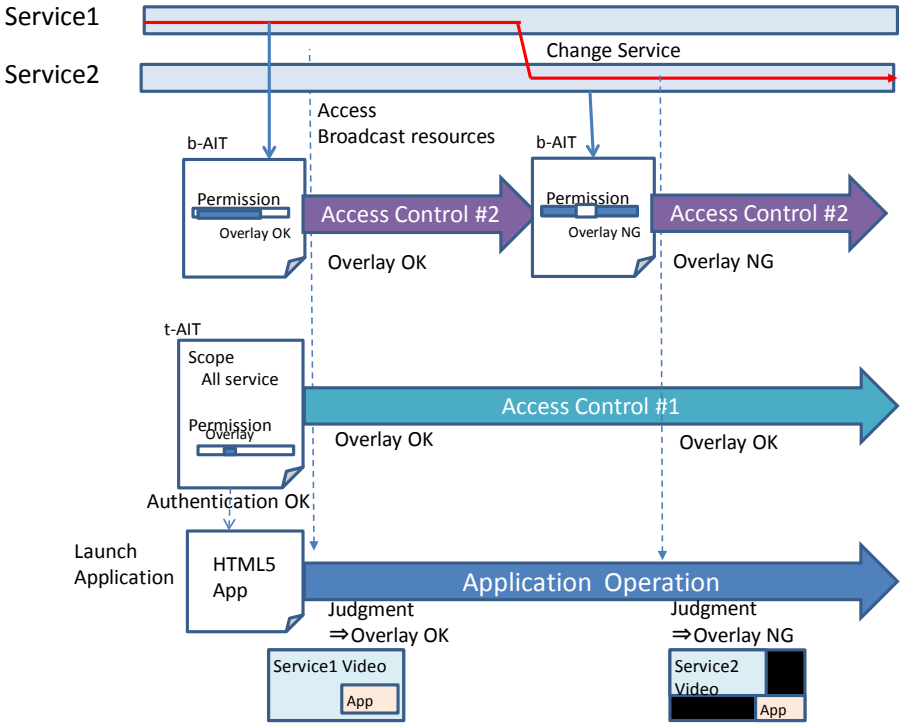


Fig. F-1 Sequence of Scenario 1

■ Scenario 2: Access control with an area specified

① The user selects and launches a non-broadcast-oriented managed application using a launcher. In this example, the application concerned is a widget class application.

② The terminal fetches the t-AIT, and verifies the application authentication. If the verification is successful, it fetches the application and launches it.

③ The terminal also fetches the b-AIT being sent in the broadcast service selected, applies an “AND” operation to the access permission bitmap specified in the external application control descriptor and the access permission bitmap specified in the broadcastPermission element of the t-AIT, and uses the result as access control information while application is running. If the b-AIT specifies that overlay is permitted (or inhibited), it is checked whether the area where overlay is permitted is large enough, based on the appSize element of the t-AIT. In this example, it is checked whether overlay display (or without an area specified) is permitted based on the b-AIT and t-AIT of Service1, and in addition it is checked whether the display area whose size is specified in the t-AIT can fit in the area where overlay is permitted (P_area1).

④ When the terminal launches the application, it determines whether access to broadcast resources is permitted based on the access control in ③, and reflects the result in the execution of the application. In this example, overlay of the application display area of the specified size on the specified area of the broadcast video of Service1 is permitted. Therefore, the permitted application is displayed on the display area of the broadcast video.

⑤ The user switches from Service1 to Service2.

⑥ The terminal performs channel selection, and begins to receive and play the broadcast stream of the newly selected service. At the same time, the terminal fetches the b-AIT associated with the newly selected service. As in ③, it applies an “AND” operation to the access permission bitmap of the t-AIT and the access permission bitmap of the b-AIT. If the b-AIT specifies that overlay is permitted (or inhibited), it is checked whether the area where overlay is permitted is large enough based on the appSize element of the t-AIT. The result is used as new access control information. In this example, the access permission bitmap of Service 2 permits access but even the minimum application display area is not small enough to fit into the area where overlay is permitted. As a result, the control information indicates that overlay display is not permitted.

⑦ Using the above control information, access by the application to broadcast resources is reevaluated, and the result is reflected in access control. In this example, overlay display is not permitted. As a result, the application is displayed outside the display area of the broadcast video.

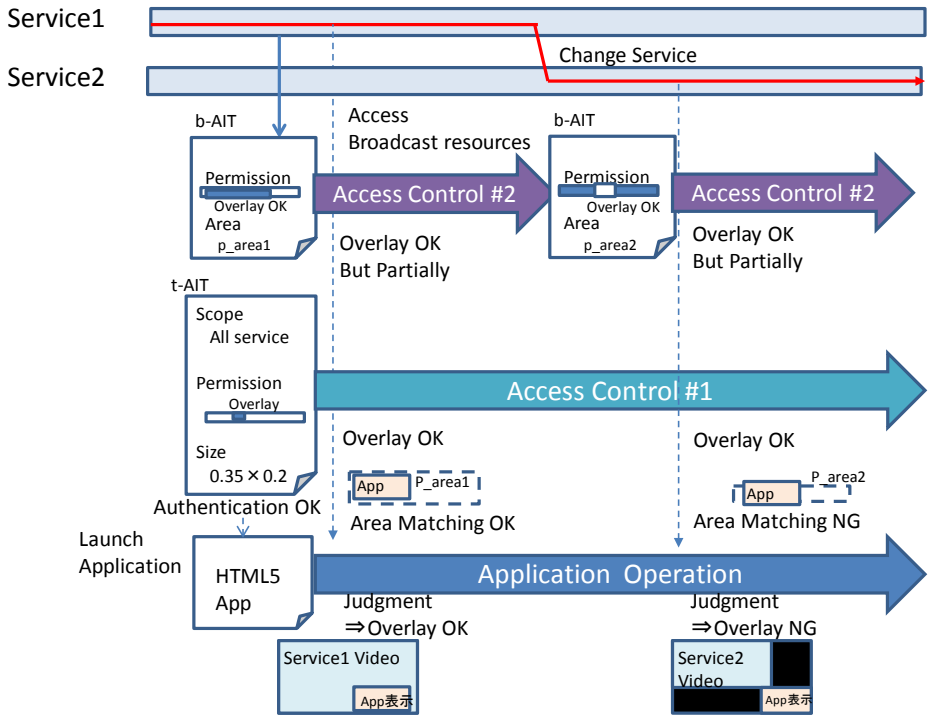


Fig. F-2 Sequence of Scenario 2

Appendix G Application Launch Control in Playing a Recorded Video

When an application is to be launched in association with the playing of a recorded video, the application to be launched may vary depending on whether a live video or a previously recorded video is to be played. In light of this, it is assumed that following descriptors used in the application information tables will be added to ARIB STD-B24 Volume 4 in order to control the application launch at the time when a recorded video is played. Application information tables that contain these descriptors are broadcast by broadcasters, and the content of the tables is stored in a video recording device. When a recorded video is to be played, this content is referred to to launch the appropriate application. The application that is launched in association with the playing of a recorded video is determined by referring to the recorded video playing application control descriptor, as specified in G.1 and the simple recorded video playing application control descriptor, as specified in G.2. If these are absent, the application indicated in the application control descriptor and the simple application location descriptor is launched.

G.1 Recorded video playing application control descriptor

In order to resolve the location from which an application associated with the playing of a recorded video can be fetched, it is assumed that this descriptor is used together with a simple video playing application location descriptor, as specified in G.2. This descriptor is placed in the application information descriptor loop of the application information table at the time of broadcasting. The transport protocol descriptor that is referred to by this descriptor was fetched at the time when the video was recorded.

Table G-1 Recorded video playing application control descriptor

Data structure	Number of bits	Bit string notation
playback_application_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
application_profile_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(m = 0; m < M; m++) {		uimsbf
transport_protocol_label	8	uimsbf
}		
}		

In specifying an application using this descriptor, it is assumed that the recorded video to be played is equivalent to a service. Therefore, in the following, “current service” indicates a piece of content that is being played (including data that is multiplexed on the provided service).

Semantics:

descriptor_tag (descriptor tag). “XX” is specified to indicate this descriptor.

application_profiles_length (application profile information length): Indicates the length in bytes of the entire application profile information contained in a loop that follows. This is the same as that of the field of the application descriptor of the same name.

application_profile (application profile): Indicates the application profile of the receiver that can execute this application. Having this profile is an indication that the receiver can execute this application. The content of the profile is defined for each application type. This is the same at that of the field of the application descriptor of the same name.

version.major (major version): Indicates the major version of the above profile. This is the same at that of the field of the application descriptor of the same name.

version.minor (minor version): Indicates the minor version of the above profile. This is the same at that of the field of the application descriptor of the same name.

version.micro (micro version): Indicates the micro version of the above profile. This is the same at that of the field of the application descriptor of the same name.

The above four fields constitute the minimum profile for executing this application. The receiver shall launch this application if this application profile information contains at least one profile for which the result of the following logical operation is true.

(Application’s profile ∈ set of profiles implemented in the terminal)

AND { (Application’s major version ＜ major version of the terminal that meets this profile)

OR [(Application’s major version ＝ major version of the terminal that meets this profile)

AND ({ Application’s minor version < minor version of the terminal that meets this profile }

OR { [application's minor version ＝ minor version of the terminal that meets this profile]

AND [application's micro version ≦ micro version of the terminal that meets this profile]})] }

service_bound_flag (service boundary flag): Indicates whether this application is valid in the current service. This is the same at that of the field of the application descriptor of the same name. If the value of this flag is "1," the application is associated with the current service only. Therefore, if the service is switched to another service, the application is terminated.
visibility (visibility): Indicates whether this application is visible to the user or to other applications when it is running. This is the same at that of the field of the application descriptor of the same name.

Value of visibility	Semantics
00	This application is invisible. However, it may produce an error report, such as outputting its log.
01	This application is invisible to the user, but is visible to other applications via an API, etc.
10	Reserved for future use
11	This application is visible to both the user and other applications.

application_priority (application priority): Indicates the relative priority of an application when multiple applications are running. This is the same at that of the field of the application descriptor of the same name.
transport_protocol_label (transport protocol label): Indicates a value that uniquely identifies the transport protocol used to transmit the application. This corresponds to a field of the same name in the recorded transport protocol descriptor. This is the same at that of the field of the application descriptor of the same name.

G.2 Simple recorded video playing application location descriptor

The purpose of the simple application location descriptor is to indicate details of the location from which an application associated with the playing of a recorded video can be fetched. One descriptor is placed for each application associated with the playing of a recorded video in the application information descriptor loop of the application information table.

Table G-2 Structure of the simple recorded video playing application location descriptor

Data structure	Number of bits	Bit string notation
playback_simple_application_location_descriptor(){		

descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for(i =0; i < N; i++){		
initial_path_bytes	8	uimsbf
}		
}		

Semantics:

descriptor_tag (descriptor tag). “XX” is specified to indicate this descriptor.

initial_path_bytes (application URL). This is a character string that indicates the URL of the entry point of the associated application. It is indicated as a relative path for which the root is the location from which the application indicated by the transport protocol descriptor can be fetched.

G.3 Application expiration date descriptor

This descriptor shows the expiration date until which the application associated with the playing of a recorded video is permitted to be launched. If this date has passed, the application is not launched. If this descriptor is to be used, it is placed in the application information descriptor loop of the application information table. No expiration date shall be set for an application whose application information descriptor loop does not have this descriptor.

Table G-3 Structure of the application expiration date descriptor

Data structure	Number of bits	Bit string notation
application_expiration_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
expiration	40	bslbf
}		

Semantics:

descriptor_tag (descriptor tag). “XX” is specified to indicate this descriptor.

expiration (expiration date and time): Indicates the expiration date of the application. This is expressed in Modified Julian Day (MJD) and Japan Standard Time (JST). T first 16 bits contain the last 16 bits of MJD, and the subsequent 24 bits express JST encoded with six 4-bit binary-coded decimal numbers (BCD).

G.4 Specification in XML format

G.4.1 Application element

The ApplicationExpireDescriptor element is added to the Application element to add a function to indicate the expiration date for an application. The syntax of the Application element is shown in TableG-4, and the structure of the Application element is shown in Fig.G-1.

Table G-4 Syntax of the Application element

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

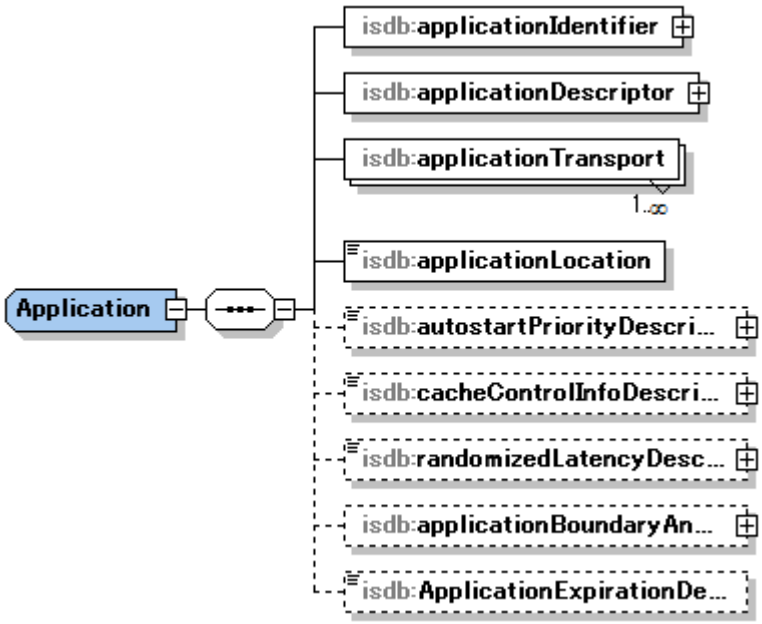


Fig.G-1 Structure of the Application element

G.4.2 ApplicationOnPlayback element

The ApplicationOnPlayback element is specified as an element that controls the application that is launched when a recorded video is to be played. Its basic structure is the same as that of the Application element. It differs from the latter in that the ApplicationDescriptor element, the applicationIdentifier element, the applicationTransport element, the applicationLocation element, and the ApplicationExpirationDescriptor element can be omitted. The ApplicationOnPlayback element can be omitted in the ApplicationList element, which is a parent element. Omission of the ApplicationOnPlayback element means that an application associated with the playing of a recorded video is not specified. If this element is specified, but the ApplicationDescriptor element, the applicationIdentifier element, the applicationTransport element, the applicationLocation element, and the ApplicationExpirationDescriptor element are omitted, the launch of the application associated with the playing of a recorded video is suppressed. The syntax of this element is shown in TableG-5, and the structure of this element is shown in Fig.G-2. The structure of the ApplicationList element, which is the parent element, is shown in Fig.G-3.

Table G-5 Syntax of the ApplicationOnPlayback element

```
<xsd:complexType name="ApplicationOnPlayback">
    <xsd:sequence>
        <xsd:element name="applicationIdentifier" type="isdb:ApplicationIdentifier"
                minOccurs="0"/>
```

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

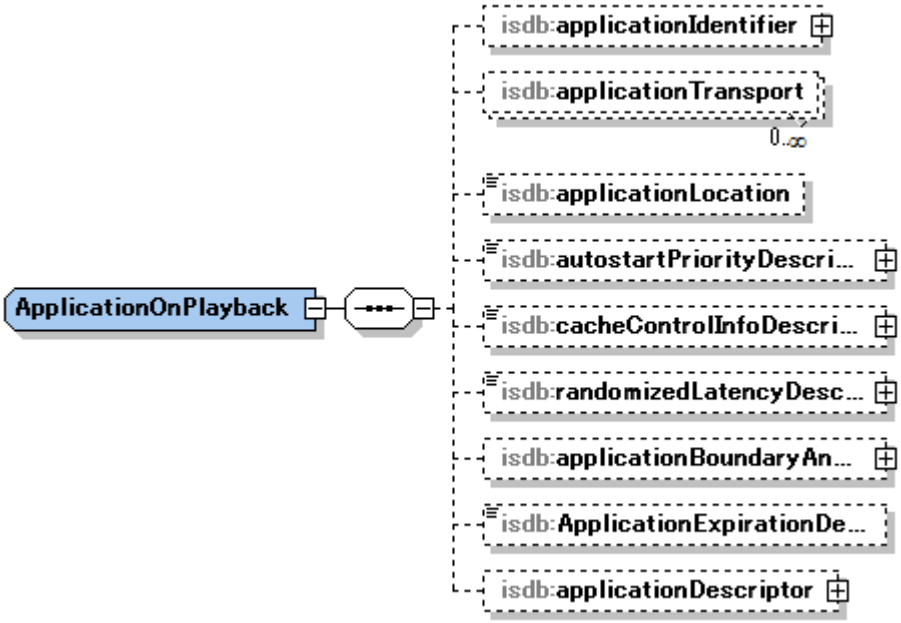
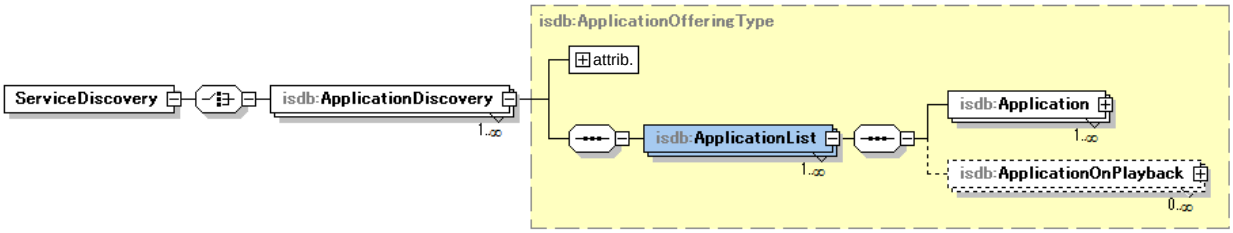


Fig.G-2 Structure of the ApplicationOnPlayback element



属性	Attribute

Fig.G-3 Structure of the ApplicationList element

G.4.3 ApplicationExpirationDescriptor element

The ApplicationExpirationDescriptor element is specified to indicate the expiration date of an application. This element has an expire attribute, in which the expiration date of the application concerned is written. Absence of this element in its parent element means that the expiration date is not defined. The syntax of this element is shown in TableG-6, and the structure of this element is shown in Fig.G-4.

Table G-6 Syntax of the ApplicationExpirationDescriptor element

```
<xsd:complexType name="ApplicationExpirationDescriptor">
    <xsd:attribute name="expire" type="xsd:dateTime"/>
</xsd:complexType>
```

Fig.G-4 Structure of the ApplicationExpirationDescriptor element

Appendix H Examples of Authentication Sequence when a Non-broadcast-oriented Managed Application is running

Examples of authentication sequence used to check the source and authenticity of a non-broadcast-oriented managed application when a service is provided using this application are shown in (1) to (3). These three sequence examples correspond to Methods 1 to 3 described in 13.5.1.

(1) Method using SSL/TLS (Method 1)

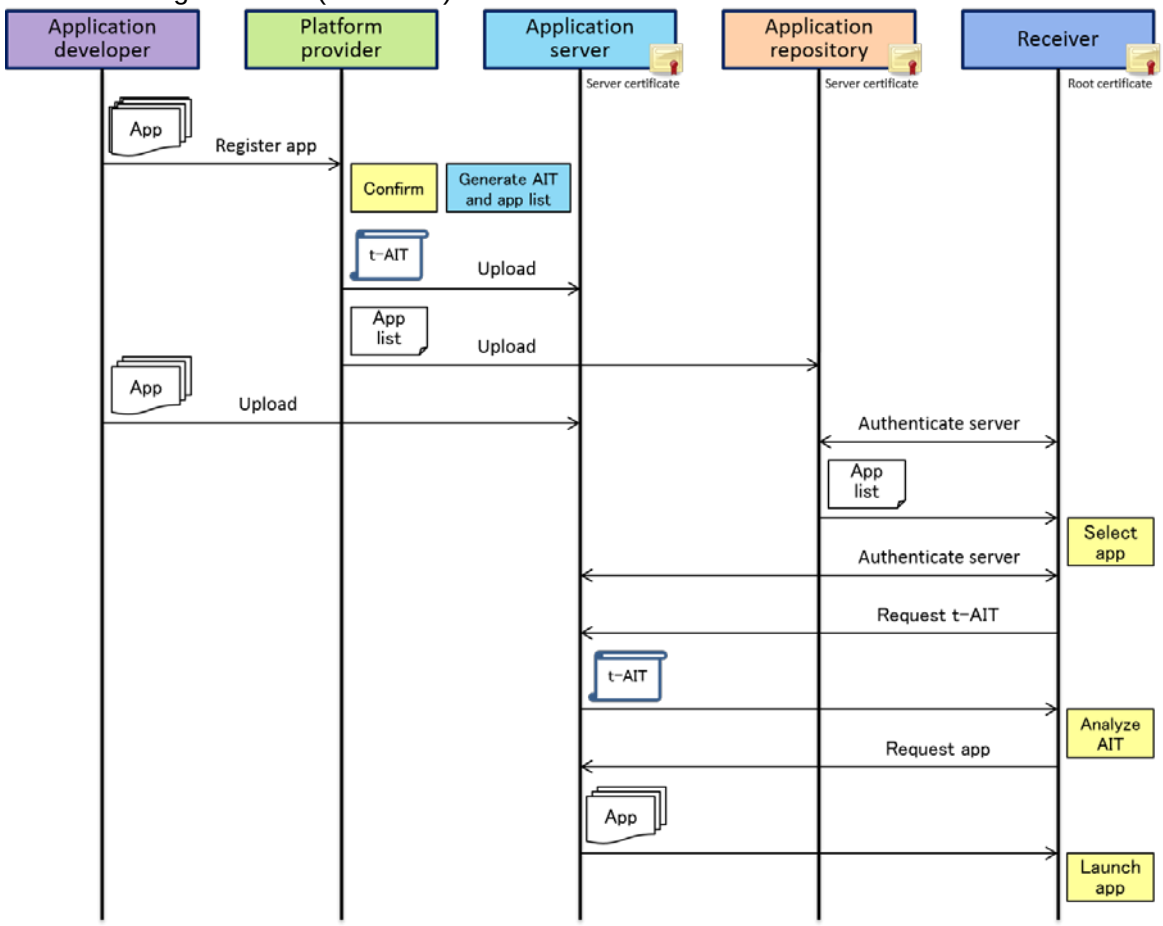


Fig. H-1 Basic sequence of application authentication (Method 1)

(2) Method using an electronic signature (Method 2)

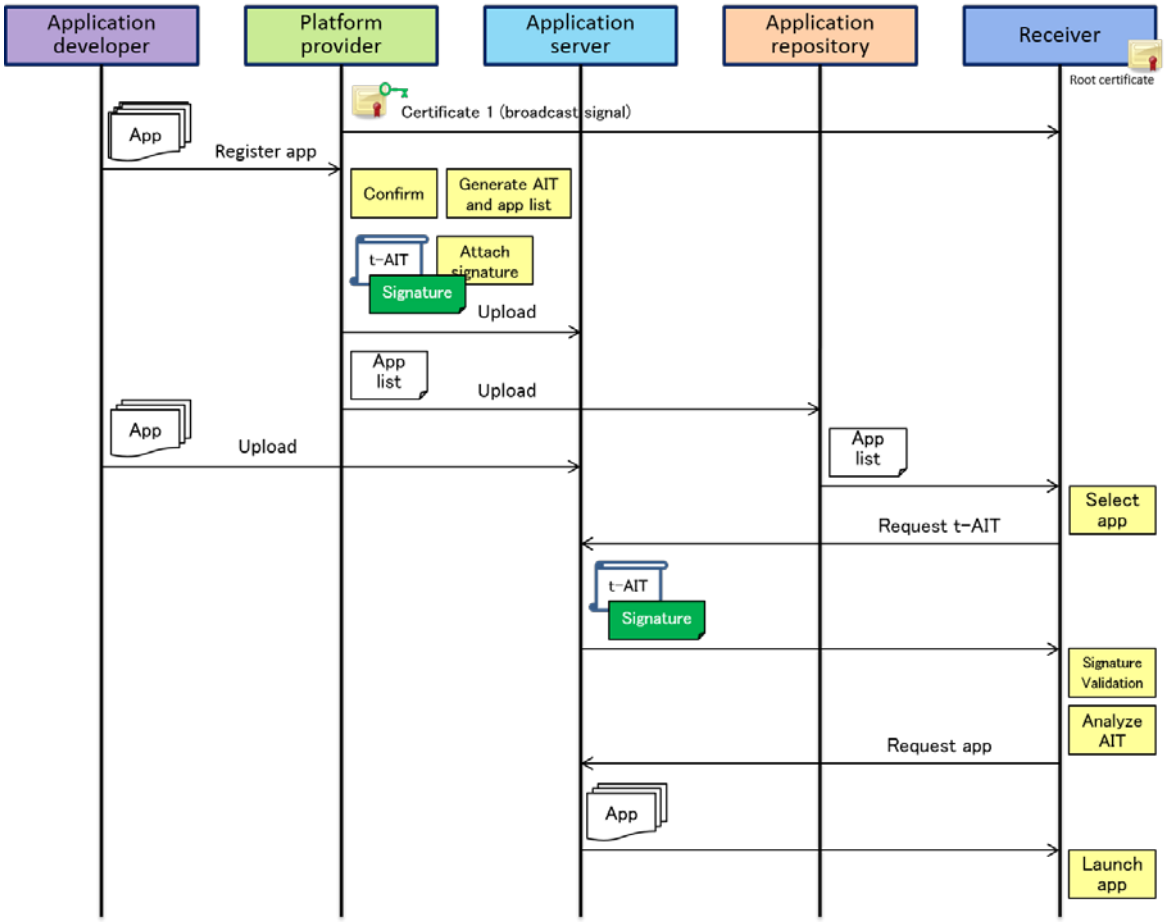


Fig. H-2 Basic sequence of application authentication (Method 2)

(3) Method using an electronic signature (Method 3)

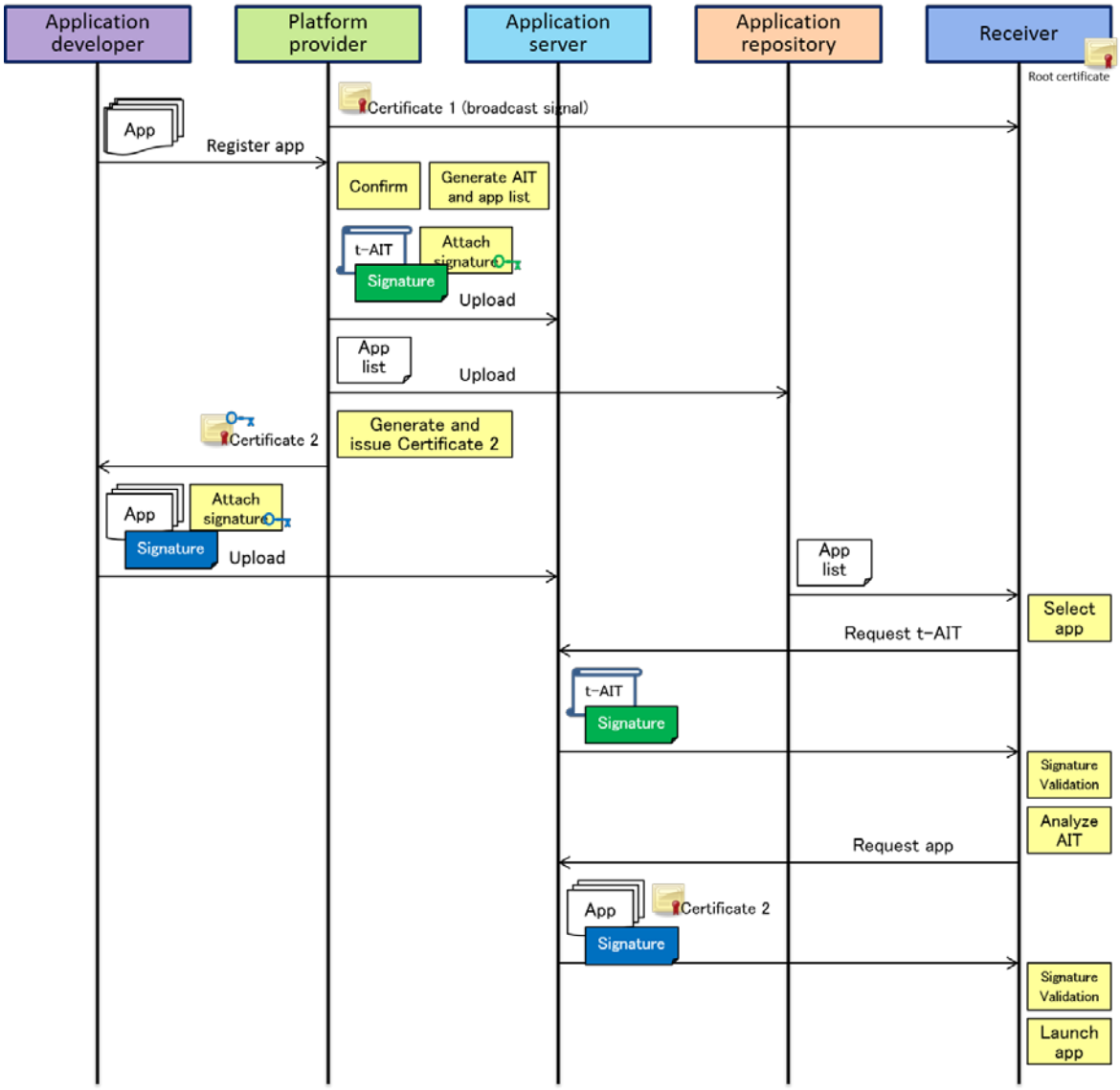


Fig. H-3 Basic sequence of application authentication (Method 3)

IPTV Standard
Integrated Broadcast-Broadband System Specification
IPTVFJ STD-0010 Version 2.0

Created on March 22, 2013: Version 1.0
Revised on June 29, 2014: Version 2.0

IPTV Forum Japan
Futaba Akasaka Bldg. 3F, 8-5-43,
Akasaka, Minato-ku, Tokyo 107-0052, Japan
Phone: 03-5858-6685
Fax : 03-5858-6675
e-mail : sec@iptvforum.jp