ATSC Standard: Companion Device

Doc. A/338:2017
17 April 2017

Advanced Television Systems Committee
1776 K Street, N.W.
Washington, D.C. 20006
202-872-9160

The Advanced Television Systems Committee, Inc., is an international, non-profit organization developing voluntary standards for digital television. The ATSC member organizations represent the broadcast, broadcast equipment, motion picture, consumer electronics, computer, cable, satellite, and semiconductor industries.

Specifically, ATSC is working to coordinate television standards among different communications media focusing on digital television, interactive systems, and broadband multimedia communications. ATSC is also developing digital television implementation strategies and presenting educational seminars on the ATSC standards.

ATSC was formed in 1982 by the member organizations of the Joint Committee on InterSociety Coordination (JCIC): the Electronic Industries Association (EIA), the Institute of Electrical and Electronic Engineers (IEEE), the National Association of Broadcasters (NAB), the National Cable Telecommunications Association (NCTA), and the Society of Motion Picture and Television Engineers (SMPTE). Currently, there are approximately 150 members representing the broadcast, broadcast equipment, motion picture, consumer electronics, computer, cable, satellite, and semiconductor industries.

ATSC Digital TV Standards include digital high definition television (HDTV), standard definition television (SDTV), data broadcasting, multichannel surround-sound audio, and satellite direct-to-home broadcasting.

Note: The user's attention is called to the possibility that compliance with this standard may require use of an invention covered by patent rights. By publication of this standard, no position is taken with respect to the validity of this claim or of any patent rights in connection therewith. One or more patent holders have, however, filed a statement regarding the terms on which such patent holder(s) may be willing to grant a license under these rights to individuals or entities desiring to obtain such a license. Details may be obtained from the ATSC Secretary and the patent holder.

Implementers with feedback, comments, or potential bug reports relating to this document may contact ATSC at https://www.atsc.org/feedback/.

Revision History

Version	Date
Candidate Standard approved	2 December 2015
Standard approved	17 April 2017
Reference to A/331 [3] editorially updated to point to the current published version	3 January 2018
Reference to A/344 [11] editorially updated to point to the current published version	3 January 2017
Reference to A/360 [10] editorially updated to point to the current published version	9 January 2018

Table of Contents

Index of Tables and Figures

ATSC Standard:
Companion Device

1. SCOPE

This document specifies the communication protocol between an ATSC 3.0 primary device and an ATSC 3.0 companion device. In this context the primary device is the primary receiver and is used to present the primary content. The companion device communicates with the primary device to present related, supplementary content, or even the same content as that being presented on the primary device. Examples of primary devices include television sets, set-top/converter boxes, and mobile devices that are capable of receiving ATSC 3.0 services. Examples of companion devices are laptops, tablets and smartphones. Use of one of these example companion devices as an ATSC 3.0 receiving device displaying primary content is out of scope of this document, such using a tablet that has a built-in ATSC 3.0 antenna as a primary viewing device. Use of a companion device to access television-related content, but not in conjunction or communication with a primary receiving device is also out of scope of this document.

1.1 Introduction and Background

This document describes communication protocols that can enable a wide variety of companion (aka second) screen user experiences. Examples of applications for this specification can be found in Annex B

1.2 Organization

This document is organized as follows:

- Section 1 – Outlines the scope of this document and provides a general introduction.
- Section 2 – Lists references and applicable documents.
- Section 3 – Provides a definition of terms, acronyms, and abbreviations for this document.
- Section 4 – System overview
- Section 5 – Specification
- Annex A – Schema
- Annex B – Usage Scenarios

2. REFERENCES

All referenced documents are subject to revision. Users of this Standard are cautioned that newer editions might or might not be compatible.

2.1 Normative References

The following documents, in whole or in part, as referenced in this document, contain specific provisions that are to be followed strictly in order to implement a provision of this Standard.

[1] IEEE: "Use of the International Systems of Units (SI): The Modern Metric System," Doc. SI 10-2002, Institute of Electrical and Electronics Engineers, New York, N.Y.

[2] ATSC: "ATSC Standard: Service Announcement (A/332)," Doc. A/332:2017, Advanced Television Systems Committee, 16 March 2017.

[3] ATSC: "ATSC Standard: Signaling, Delivery, Synchronization and Error Protection," Doc. A/331:2017, Advanced Television Systems Committee, Washington, D.C., 6 December 2017.

[4] ETSI: "Hybrid Broadcast Broadband TV – HbbTV," Document ETSI TS 102 796 v1.4.1, European Telecommunications Standards Institute, European Broadcasting Union, August, 2016. http://www.etsi.org/deliver/etsi_ts/102700_102799/102796/01.04.01_60/ts_102796v010401p.pdf.

[5] IETF: RFC 2616, "HypertextTransfer Protocol — HTTP/1.1," Internet Engineering Task Force, Reston, VA, June, 1999. http://tools.ietf.org/html/rfc2616.

[6] IETF: RFC 6455, "The WebSocket Protocol," Internet Engineering Task Force, December, 2011. http://tools.ietf.org/html/rfc6455.

[7] "The WebSocket API", W3C, https://www.w3.org/TR/websockets/

[8] IETF: RFC 7231, "HypertextTransfer Protocol (HTTP/1.1): Semantics and Content," Internet Engineering Task Force, June, 2014. http://tools.ietf.org/html/rfc7231.

[9] IETF: BCP 47, "Tags for Identifying Languages," Internet Engineering Task Force, Reston, VA, September 2009. https://tools.ietf.org/html/bcp47

[10] ATSC: "ATSC Standard: ATSC 3.0 Security and Service Protection," Doc. A/360:2018, Advanced Television Systems Committee, 9 January 2018.

2.2 Informative References

The following documents contain information that may be helpful in applying this Standard.

[11] ATSC: "ATSC Standard: Interactive Content," Doc. A/344:2017, Advanced Television Systems Committee, 18 December 2017.

3. DEFINITION OF TERMS

With respect to definition of terms, abbreviations, and units, the practice of the Institute of Electrical and Electronics Engineers (IEEE) as outlined in the Institute's published standards [1] shall be used. Where an abbreviation is not covered by IEEE practice or industry practice differs from IEEE practice, the abbreviation in question will be described in Section 3.3 of this document.

3.1 Compliance Notation

This section defines compliance terms for use by this document:

shall – This word indicates specific provisions that are to be followed strictly (no deviation is permitted).

shall not – This phrase indicates specific provisions that are absolutely prohibited.

should – This word indicates that a certain course of action is preferred but not necessarily required.

should not – This phrase means a certain possibility or course of action is undesirable but not prohibited.

3.2 Treatment of Syntactic Elements

This document contains symbolic references to syntactic elements used in the subsystems described herein. These references are typographically distinguished by the use of a different font (e.g., restricted), may contain the underscore character (e.g., sequence_end_code) and may consist of character strings that are not English words (e.g., dynrng).

3.2.1 Reserved Elements

One or more reserved bits, symbols, fields, or ranges of values (i.e., elements) may be present in this document. These are used primarily to enable adding new values to a syntactical structure without altering its syntax or causing a problem with backwards compatibility, but they also can be used for other reasons.

The ATSC default value for reserved bits is '1.' There is no default value for other reserved elements. Use of reserved elements except as defined in ATSC Standards or by an industry standards-setting body is not permitted. See individual element semantics for mandatory settings and any additional use constraints. As currently-reserved elements may be assigned values and meanings in future versions of this Standard, receiving devices built to this version are expected to ignore all values appearing in currently-reserved elements to avoid possible future failure to function as intended.

3.3 Acronyms and Abbreviation

The following acronyms and abbreviations are used within this document.

ATSC – Advanced Television Systems Committee
AEA – Advanced Emergency Alert
EAM – Emergency Alert Message
ESG – Electronic Service Guide
CD – ATSC 3.0 Companion Device
HbbTV – Hybrid Broadcast Broadband Television
HTML5 – Hyper Text Markup Language 5
HTTP – HyperText Transfer Protocol
JSON – JavaScript Object Notation
PD – ATSC 3.0 Primary Device
SSDP – Simple Service Discovery Protocol

3.4 Extensibility

The message structure used for HTTP protocol provides extensibility based on supported mechanisms in that protocol. The message structure used for WebSocket provides extensibility based on message version field (`PDCDMessageVersion` in Section 5.6.1.3).

4. SYSTEM OVERVIEW

This document specifies the communication protocol between a PD and a CD.

5. SPECIFICATION

5.1 System Architecture

There are several device models described below including launching a CD app, PD to CD app communication and CD app to PD communication. Cross-Origin requests as described in clause 14.8 of [4] are supported.

5.2 Device Model

5.2.1 Launching a Companion Device Application

The architecture for launching a companion device application is illustrated in Figure 5.1.

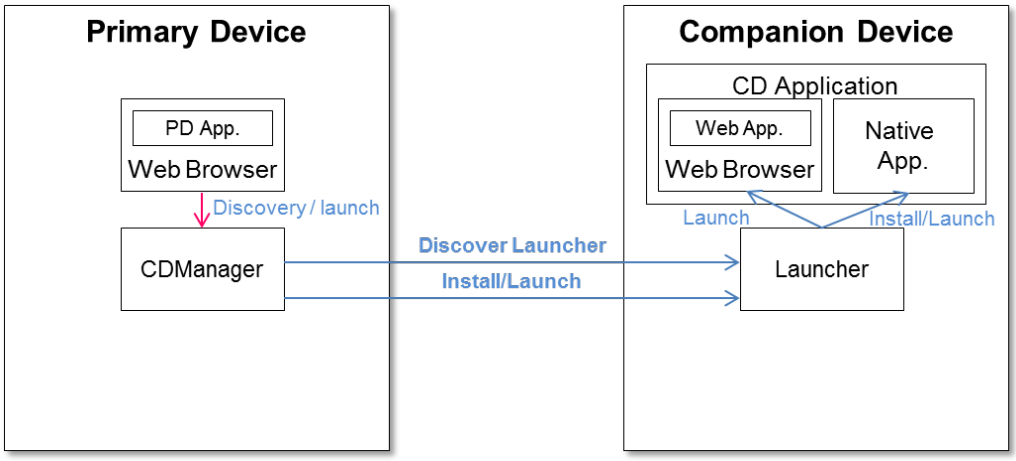

Figure 5.1 Architecture for launching a companion device application.

The following functions are distinguished in this architecture:

- Web Browser: running the PD application consists of using HTML5 and associated web technologies. The PD application is a Broadcaster Application. Both the Broadcaster Application and the application environment are described in A/344, Interactive Content [11].
- CDManager: resides in the PD. The CDManager is responsible for discovering the CDs with running Launchers and sending application launch/install information to those Launchers. Section 5.8 shall apply.
- Launcher: resides in the CD. The Launcher is responsible for communicating with the CDManager of the PD and launching and/or installing the CD application. The requirements on the Launcher shall be as described in clause 14.3 of HbbTV [4].

To launch a CD application, the `launchCSApp` method of Section 5.8 shall be used. The payload as given in clause 14.4.2.1 of HbbTV [4] shall specify a WebSocket server endpoint.

An example of such a payload for a WebSocket endpoint is:

```
{
“launch” : [
{“launchURL” : “https://www.examples-r-us.com/quiz-fallback-application.html?
colour=blue&application_uri=ws://192.168.11: 992/atsc/”, “applicationType” :
“native” } ,
   ]
}
```

The protocol for launching a CD application from a CDManager is as follows. The CDManager requests the launch of the CD application by sending an HTTP POST request to the Application-URL of the Launcher. The Application-URL of the Launcher is obtained from using the discovery in Section 5.3. The BODY data of the HTTP POST request shall contain the payload of the `launchCSApp` method, indicating the CD application to be launched (see clause 14.4.2.2 of HbbTV [4]).

5.2.2 Application to Application Communication

The architecture for application-to-application communication is illustrated in Figure 5.2.

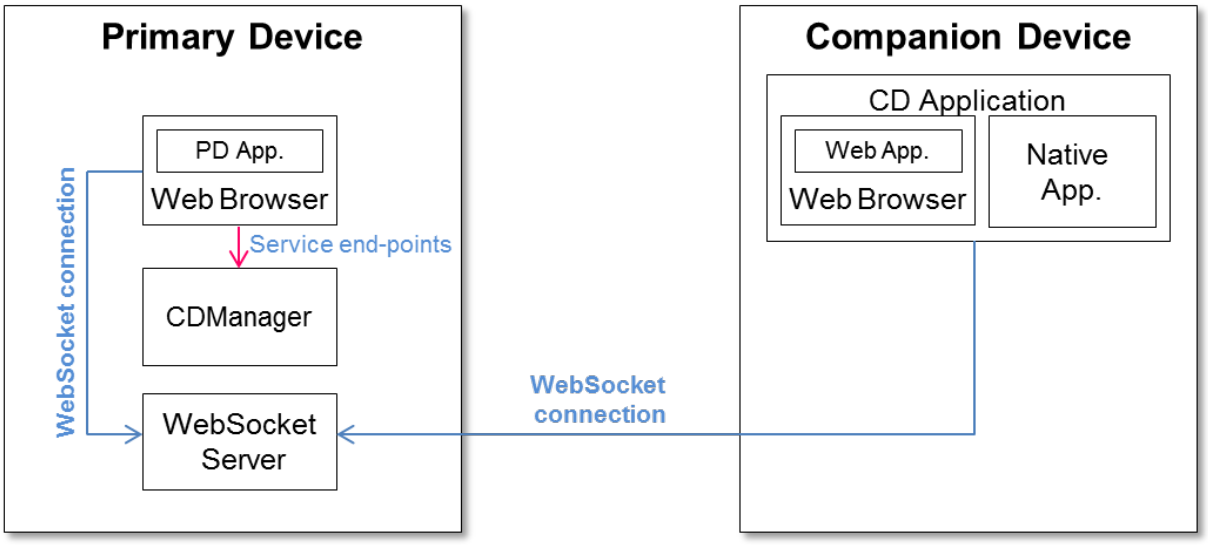

Figure 5.2 Architecture for application-to-application communication.

The following functions are distinguished in this architecture:

- CDManager: responsible for providing service endpoints for application-to-application communication. The `CDManager` API shall be as defined in Section 5.8.
- WebSocket Server: resides in the PD. WebSocket Server is responsible for handling web socket connections from PD applications and from CD applications. WebSocket protocol is defined in [6] and W3C API is defined in [7]. Either 'ws' or 'wss' or both of these URI schemes shall be supported as documented in RFC 6455 [6]. The communication between the PD applications and CD applications shall be as described in clause 14.5 of HbbTV [4] with the exception that with respect to clause 14.5.4 of HbbTV [4] "wss:" scheme is supported in addition to "ws:" scheme.

Applications should not use both "ws:" and "https:" scheme in the same application.

If a CD application has been launched by a PD application then the location of the service endpoint shall have been provided to the CD application as one of its launch parameters in the `launchCSApp` method payload. This endpoint shall be the remote endpoint of a WebSocket server.

5.2.3 Companion Device Application to Primary Device Communication

The architecture for CD application to PD communication is illustrated in Figure 5.3.

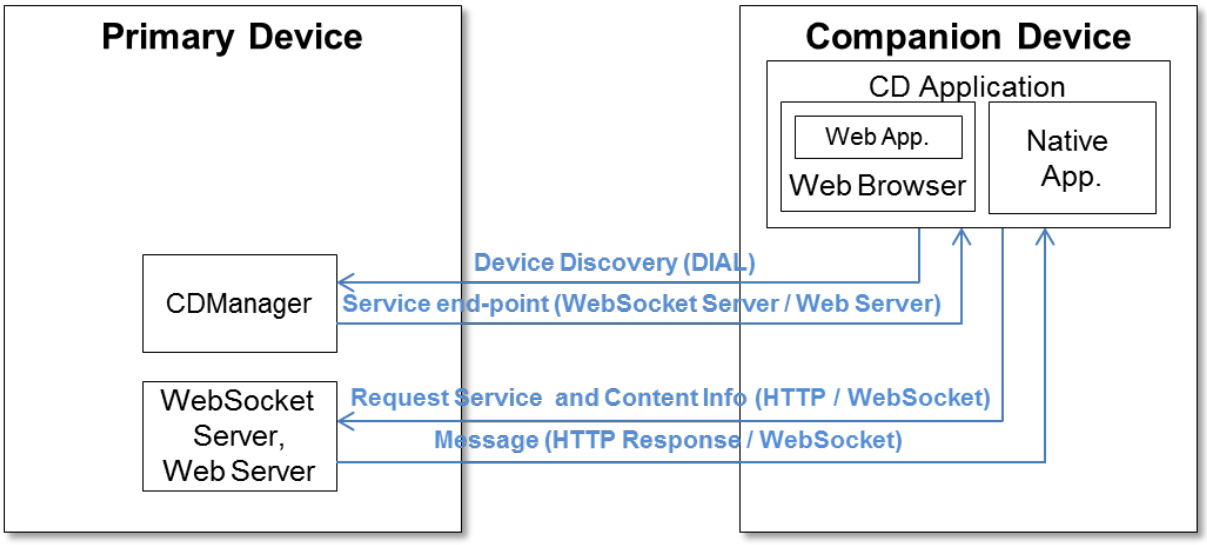

Figure 5.3 Architecture for CD application to PD communication.

- CDManager: resides in the PD. The CDManager shall be responsible for responding to discovery requests from CD applications and for providing the service endpoints of its Web Server and WebSocket Server.
- Web Server: resides in the PD. The Web Server shall be responsible for handling HTTP requests from CD applications and responding with the service and content information of the PD.
- WebSocket Server: resides in the PD. The WebSocket Server shall be responsible for handling WebSocket connections from the CD application and responding with the service and content information of the PD.
- CD Application: resides in the CD. The CD application shall be responsible for discovering the PD and obtaining the service and content information of the PD via HTTP and WebSocket protocols.

A CD application may establish communication with servers providing services on a PD. In doing so, the CD application shall first discover the PD and in the process obtain the remote endpoints of its Web Server and WebSocket Server. At this point, the CD application may obtain service and content information through the WebSocket Server or via an HTTP GET request. In the former case, the CD application shall first establish a WebSocket connection and then send a request for service and content information through this connection. In the latter case, the CD application shall issue an HTTP GET and receive an HTTP response. If an encrypted connection is established between a CD application and PD for information exchange, then methods defined in Section 5.6 of A/360 [10] shall be used.

5.3 Protocol for Discovery

Both the PD and the CD app are capable of sending multicast discovery messages searching and/or advertising their presence and ATSC 3.0 PD-CD service support.

It is possible that a household has more than one PD on the home network, so a CD application could receive discovery messages from multiple PDs. In that case, the CD application can ask the

user which one(s) to interact with (displaying information from the discovery messages to help the user decide). The converse is also true in that there may be more than one CD on the home network.

The following mechanisms are supported for discovery.

5.3.1 CD App Multicast Search Request Message for Discovering PD and Response from PD

This request shall be performed as described in Section 14.7 of HbbTV [4] and further modified as described below.

5.3.1.1 Introduction

In the situation where a CD application has been launched by a PD application, information regarding the location of the service endpoints exposed by the PD may be conveyed as parameters in the `launchURL` as described in Section 5.2.1.

However, for "Companion Device application to Primary Device communication" as described in Section 5.2.3, the CD application needs to be able to discover the locations of the Web Server and WebSocket Server endpoints of the PD. The methods for achieving this shall be as described in Section 5.3.1.2. An example is provided in Section 5.3.1.3.

5.3.1.2 PD and Service Endpoint Discovery

In discovering the PD and service endpoint, clause 14.7.2 of HbbTV [4] shall apply only.

The Web Server endpoint URL for "Companion Device Application to Primary Device" communication as given in Section 5.2.3 shall be the Application Resource URL for HbbTV [4] except that the WebSocket Server endpoint URL shall be the `<X_ ATSC_App2AppURL>` element.

5.3.1.3 Discovery Example

This is an example of discovering the PD and its service endpoints from a CD application as described in Section 5.2.3.

Device Discovery Request:

The CD application initiates a device discovery by performing an M-SEARCH using the SSDP protocol with the Search Target header (ST) as defined below and further as described in clause 14.7 of HbbTV [4]:

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: <seconds to delay response>
ST: urn:schemas-atsc.org.device: companionDevice:1.0
```

Discovery Response:

The PD responds with an `HTTP/1.1 OK` and `LOCATION` header, and `ST`:

```
HTTP/1.1 200 OK
CACHE-CONTROL: max-age = <seconds until advertisement expires>
EXT:
LOCATION: <URL for UPnP description for root device>
SERVER: <OS/version UPnP/1.0 product/version>
ST: urn: schemas-atsc.org.device: primaryDevice:1.0
USN: <advertisement UUID>
```

Device Description Request:
The CD application requests the device description file using an HTTP GET request to the `LOCATION` URL:

```
GET <path component of the LOCATION URL> HTTP/1.1
Origin: http://cs.services.broadcaster.com/
```

Device Description Response:
The PD responds with an `HTTP/1.1 OK` header containing the `Application-URL` as described in clause 14.7 of HbbTV [4]:

```
HTTP/1.1 200 OK
CONTENT-LANGUAGE: <language used in description>
CONTENT-LENGTH: <bytes in body>
CONTENT-TYPE: text/xml; charset="utf-8"
Application-URL: http:// xx.xx.xx.xx:yyyy/applications
Access-Control-Allow-Origin:*
```

The `Application-URL` is used as the Web Server endpoint of the PD.

Application Information Request:
As in the Device Description Response example, the REST service is on an `Application-URL` of `http://xx.xx.xx.xx:yyyy/applications` and the following are examples of how the PD service endpoints are discovered.

A HTTP GET message is sent to `xx.xx.xx.xx`, port `yyyy` as follows:

```
GET /applications/ATSC HTTP/1.1
Origin: http://cs.services.broadcaster.com/4
```

Application Information Response:
An HTTP response is returned as follows

Header:
```
HTTP/1.1 200 OK
Origin: http://cs.services.broadcaster.com/
```

Body:
```
<?xml version="1.0" encoding="UTF8"?>
<service xmlns="urn:dialmultiscreenorg:schemas:dial" dialVer="1.7">
    <name>ATSC</name>
    <options allowStop="false"/>
    <state>running</state>
    <additionalData>
        <X_ ATSC_App2AppURL>URL of App2App comm. endpoint
           </X_ATSC_App2AppURL>

        <X_ ATSC_UserAgent>Value of ATSC UA header
```

```
            </X_ATSC_UserAgent>
    </additionalData>
</service>
```

<X_ATSC_App2AppURL> is used for the WebSocket endpoint of the PD.

5.3.2 PD Advertisement Message (Multicast)

When a PD joins the network and/or when it wishes to advertise itself, it shall multicast a SSDP message to advertise itself as a PD. Additionally, a PD may send advertisement multicast messages periodically. Multicast advertisement messages from a PD shall be sent to the address 239.255.255.250 and port 1900 i.e., (239.255.255.250:1900). The advertisement message shall consist of the following fields:

- A PD device type of "urn:schemas-atsc.org:device:primaryDevice:1.0" shall be signaled in a Notification Type (NT) header.
- An identifier "uuid:<device uuid>:urn:schemas-atsc.org:device:primaryDevice:1.0", which uniquely identifies this PD for the advertisement, shall be signaled in a USN (Unique Service Name) header.
- The duration for which the PD advertisement message is valid shall be signaled in a CACHE-CONTROL header.
- Additional information about the PD may be signaled in the LOCATION header.

An example multicast advertisement message from a PD is as shown below:

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age = <advertisement validity duration in seconds>
LOCATION: <URL for primary device>
NT: urn:schemas-atsc.org:device:primaryDevice:1.0
NTS: ssdp:alive
SERVER: <Primary device ID/ Version>
USN: uuid:<device uuid>:urn:schemas-atsc.org:device:primaryDevice:1.0
```

5.3.3 CD Advertisement Message (Multicast)

When a CD joins the network and/or when it wishes to advertise itself, it shall multicast a SSDP message to advertise itself as a CD. Additionally, a CD may send advertisement multicast messages periodically. Multicast advertisement messages from a CD shall be sent to the address 239.255.255.250 and port 1900; i.e., (239.255.255.250:1900). The advertisement message shall consist of following fields:

- A CD device type of "urn:schemas-atsc.org:device:companionDevice:1.0" shall be signaled in a Notification Type (NT) header.
- An identifier "uuid:<device uuid>:urn:schemas-atsc.org:device:companionDevice:1.0", which uniquely identifies this companion device for the advertisement, shall be signaled in a USN (Unique Service Name) header.
- The duration for which the CD advertisement message is valid shall be signaled in a CACHE-CONTROL header.

- Additional information about the companion device may be signaled in the `LOCATION` header.

An example multicast advertisement message from a CD is as shown below:

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age = <advertisement validity duration in seconds>
LOCATION: <URL for companion device>
NT: urn:schemas-atsc.org:device:companionDevice:1.0
NTS: ssdp:alive
SERVER: <Companion device ID/ Version>
USN: uuid:<device uuid>:urn:schemas-atsc.org:device:companionDevice:1.0
```

5.3.4 PD Search Request Message for discovering CD (Multicast)

A search from a PD for a CD on the network shall be done using the following steps:

- A SSDP multicast search M-SEARCH request shall be sent to address `239.255.255.250` and port `1900`; i.e., (`239.255.255.250:1900`).
- The multicast search M-SEARCH request shall set the `ST` header to CD device type as "`urn:schemas-atsc.org:device:companionDevice:1.0`".
- The maximum response delay in seconds within which a CD should send a response shall be indicated in the `MX` header.

An example multicast search request from a PD is shown below:

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: <max response delay in seconds>
ST: urn:schemas-atsc.org:device:companionDevice:1.0
```

5.3.5 CD Search Response Message (unicast)

When a CD receives a multicast search message from a PD as described in Section 5.3.4 it shall respond with a unicast search response to the PD search message. For this, it shall perform following steps:

- Send a unicast search response within a random duration between 0 to `<max response delay in seconds>` seconds, where `<max response delay in seconds>` is found in the `MX` header of the M-SEARCH request from the PD (Section 5.3.4).
- An XML element `<DevName>` which indicates a human-friendly CD device name may be sent in the CD search response in the message body.

An example unicast response to M-SEARCH from a CD is shown below:

```
HTTP/1.1 200 OK
CACHE-CONTROL: max-age = <advertisement validation duration in seconds>
DATE: <when response was generated>
LOCATION: <URL for device/ service description for companion device>
SERVER: <Companion device ID/ Version>
ST: urn:schemas-atsc.org:device:companionDevice:1.0
USN: uuid:<device uuid>:urn:schemas-atsc.org:device:companionDevice:1.0
```

5.4 Launching a Companion Device Application

In launching a CD application from a PD, all normative portions of clause 14.3.1 and clause 14.3.2 of HbbTV [4] shall apply, except that the HbbTV CS application shall be a CD application, the HbbTV Terminal shall be a PD and the HbbTV Terminal application shall be a PD application. In addition, the payload of the `launchCSApp` method shall either supply the remote endpoint of a WebSocket Server or the endpoint multicast address of a multicast group. The CD application may use either of these endpoints to receive information from the PD application.

5.5 Application to Application Communication

Application-to-application communication shall be through a WebSocket Server on the PD as specified in clause 14.5 of HbbTV [4] with the exception that with respect to section 14.5.4 of HbbTV [4] "wss:" scheme is supported in addition to "ws:" scheme.

5.6 Companion Device Application to Primary Device Communication

CD application to PD communication shall consist of the following steps:

- A CD application discovers an available PD and obtains its Web Server and WebSocket service endpoints as described in Section 5.3.1.2.
- A CD application requests information via HTTP or WebSocket service endpoints.
- A PD responds with information via an HTTP response or a WebSocket connection.

There are two service endpoints. HTTP uses one of the endpoints for asynchronous communications, and WebSocket uses the other of the endpoints for synchronous communications. Communications for the ESG and Service Information, Service, Show and Segment Data shall use HTTP, and communications for the Service and Content Identification, and Media Playback State shall use WebSocket. Communications for the Media Timeline shall use HTTP and/or WebSocket. If an encrypted connection is established between a CD application and PD for information exchange, then methods defined in Section 5.6 of A/360 [10] shall be used.

5.6.1 Message Structure

5.6.1.1 Message Structure for HTTP Request

The CD application sends a HTTP GET request to the PD as follows:

Request URL: `<ATSCCS_PDURL>/ServiceName?<param1=val1&…>`

`ATSCCS-PDURL` has previously been obtained during discovery procedure and shall be as described in Section 5.3.1.2.

`ServiceName` shall be as listed in Table 5.1.

Table 5.1 HTTP Request Services

ServiceName	Description	Reference Section
`atsc3.csservices.esg.1`	Electronic Service Guide	5.6.2.4
`atsc3.csservices.esg.2`	Electronic Service Guide	5.6.3.2
`atsc3.csservices.mt.1`	Media Timeline	5.6.5.2

5.6.1.2 Message Structure for HTTP Response

When the PD receives an HTTP GET request as described in Section 5.6.1.1, it shall respond with an HTTP status code and body shall be as given in **Table 5.2**.

Table 5.2 HTTP Response Fields

Field Name	Cardinality	Description
ATSCCS_Message	1	
PDServiceName	1	Name of service
MessageBody	0..1	Message body data

5.6.1.3 Message Structure for WebSocket

The Subscription Message Structure shall be as defined in Section 5.6.1.3.1 and the Notification Message Structure shall be as defined in Section 5.6.1.3.2.

5.6.1.3.1 Subscription Message Structure

Subscription related messages between PD and CD shall use the subscription message structure shown in **Table 5.3** below. The list of supported service enumeration values shall be as shown in **Table 5.4** . The list of supported message enumeration values shall be as shown in **Table 5.5**.

Table 5.3 Subscription Message Structure

Field Name	Cardlin-ality	Data type	Included in Message Type	Description
PDCDMessageVersion	1	Unsigned Integer	All	Version of this subscription message structure. The upper 6 bits shall indicate major version and lower two bits shall indicate minor version. The version of this subscription message structure shall be 0x004 i.e. version 1.0.
PDCDServiceName	1	String	All	The service name, which uniquely identifies the PD-CD service. The enumerated service name values shall be as defined in **Table 5.4**. PD and CD conforming to this version of this specification shall be capable of ignoring a message received with PDCDServiceName other than the names defined in **Table 5.4**.
PDCDMessageType	1	String	All	Identifies the type of message. The message type enumeration values shall be as defined in **Table 5.5**. Two categories of message types are defined. This includes request message types and response message types. Depending upon the message type (identified by the PDCDMessageType) the rest of the message structure contains different types of message fields.
PDCDRespCode	1	Unsigned Integer	Response Message types	A success or failure status code for the corresponding request.
PDCDSubDuration	1	Unsigned Integer	All except cancel and cancelResponse	Subscription duration. When the message is sent from CD to PD this field shall indicate the requested subscription duration. When the message is sent from PD to CD this field shall indicate the duration for which subscription is active.

Table 5.4 Service Enumeration Values

PDCDServiceName	Description
atsc3.services.esg.1	Electronic Service Guide
atsc3.services.mps.1	Media Playback State
atsc3.services.mt.1	Media Timeline

Table 5.5 Message Type Enumeration Values

Message Type Enumeration	Description	Allowed Direction for this Message Type
Request Message Types		
subscribe	Message to request a subscription	From CD to PD
cancel	Message to cancel a subscription	From CD to PD
renew	Message to renew a subscription	From CD to PD
Response Message Types		
subscribeResponse	Response Message to the subscription request	From PD to CD
cancelResponse	Response Message to the cancel request	From PD to CD
renewResponse	Response Message to the renew request	From PD to CD

The subscription-related messages from PD to CD and from CD to PD shall be sent in JSON format conforming to the JSON schema shown in Section A.1.

5.6.1.3.2 Notification Message Structure

Notification messages sent from PD to CD shall use the message structure shown in **Table 5.6** below. The supported notification service enumeration values shall be as shown in **Table 5.7**.

Table 5.6 Notification Message Structure

FieldName	Cardinality	Data type	Description
PDCDMessageVersion	1	Unsigned Integer	Version of the notification message structure. The upper 6 bits shall indicate major version and lower two bits shall indicate minor version. The version of this notification message structure shall be 0x004 i.e. version 1.0.
PDCDServiceName	1	String	The service name, which uniquely identifies the PD-CD service. The enumerated service name values shall be as defined in **Table 5.7**. PD and CD conforming to this version of this specification shall be capable of ignoring a message received with PDCDServiceName other than the names defined in **Table 5.7**.
MessageBody	0..1	Bytes	Message specific data fields. The syntax and semantics of the MessageBody shall be as defined in individual messages of individual services in Sections 5.6.2 to5.6.6.

Table 5.7 Notification Service Enumeration Values

PDCDServiceName	Description
atsc3.services.esg.1	Electronic Service Guide
atsc3.services.mps.1	Media Playback State
atsc3.services.mt.1	Media Timeline

Notification messages shall be sent only from PD to CD. These messages from PD to CD shall be sent in JSON format conforming to the JSON schema shown in Section A.2.

The following steps shall be performed by PD and CD for communication over a WebSocket to start receiving notification messages for a service.

1) A CD shall send a subscription message to a PD as specified in **Table 5.3**. For this message, the `PDCDServiceName` field shall be set to one of the services defined in **Table 5.4** under the column `PDCDServiceName` (e.g., "`atsc3.services.esg.1`" or "`atsc3.services.mps.1`") and the `PDCDMessageType` field shall be set to "subscribe". This subscription message shall include a requested subscription duration value in `PDCDSubDuration` field.
2) Upon receiving a subscription message from a CD, the PD supporting the service in the received `PDCDServiceName` field shall send a subscription message response to the CD as specified in **Table 5.3**. For this message, the `PDCDServiceName` field shall be set to the same name as in the `PDCDServiceName` field (e.g., "`atsc3.services.esg.1`" or "`atsc3.services.mps.1`") in the received subscription message by the PD and `PDCDMessageType` field shall be set to "`subscribeResponse`". In this message, the PD shall include in the `PDCDSubDuration` field a value for which the renewed subscription is valid.
3) At or before a current subscription ends at the time indicated by the `PDCDSubDuration` field, the CD which is interested in continuing to receive a notification message for the service shall send a subscription renewal message to the PD as specified in **Table 5.3**. In this message, the `PDCDServiceName` field shall be set to the same service name as used in the subscription message (in step 1 above) (e.g., "`atsc3.services.esg.1`" or "`atsc3.services.mps.1`") and the `PDCDMessageType` field shall be set to "`renew`". This subscription message shall include a requested subscription duration value for this renewal request in the `PDCDSubDuration` field.
4) Upon receiving a subscription renewal message from the CD, the PD shall send a subscription renew message response to the CD as specified in **Table 5.3**. For this message, the `PDCDServiceName` field shall be set to the same name as in the `PDCDServiceName` field (e.g., "`atsc3.services.esg.1`" or "`atsc3.services.mps.1`") in the received subscription renewal message by the PD and the `PDCDMessageType` field shall be set to "`renewResponse`". In this message, the PD shall include in the `PDCDSubDuration` field a value for which the renewed subscription is valid.
5) At any time when subscribed, the CD may send a subscription cancel message to the PD as specified in **Table 5.3**. In this message, the `PDCDServiceName` field shall be set to the same service name as used in the subscription message (in step 1 above) or in the subscription renewal message (in step 3 above) (e.g., "`atsc3.services.esg.1`" or "`atsc3.services.mps.1`") and the `PDCDMessageType` field shall be set to "`cancel`".
6) Upon receiving a subscription cancel message from the CD, if the CD is currently subscribed with this PD to receive the service corresponding to the value included in the `PDCDServiceName` field of the subscription cancel message, the PD shall send a subscription cancel message response to CD as specified in **Table 5.3**. In this message, the `PDCDServiceName` field shall be set to the same name as in `PDCDServiceName` field (e.g., "`atsc3.services.esg.1`" or "`atsc3.services.mps.1`") in the received subscription cancel message by the PD and the `PDCDMessageType` field shall be set to "`cancelResponse`".

7) Once a CD is subscribed with a PD for a particular service, the PD can send a notification message to the subscribed CD at any time. For this the PD shall use the Notification message structure as specified in **Table 5.6**. In this message, the `PDCDServiceName` field shall be set to the name of the service (e.g., "`atsc3.services.esg.1`" or "`atsc3.services.mps.1`") for which the notification is sent and for which the CD is subscribed with the PD. The `MessageBody` shall be set to the `MessageBody` as defined for the service.

5.6.2 Protocol and Message Content for Service and Content Identification Communication

5.6.2.1 Protocol

WebSocket (Notification)

5.6.2.2 Message Content

The message content of the Service and Content Identification communication message shall be as specified in Section 5.6.1.3.2.

The service and content identification communication message shall be JSON formatted and the `MessageBody` shall conform to the JSON schema shown in Annex A, Section A.3.

Table 5.8 describes the structure of the service and content identification communication message in a more illustrative way. The "Description" column in **Table 5.8** gives the semantics of this message's fields.

Table 5.8 Message Content for Service and Content Identification

FieldName	Cardinality	Data type	Description
`MessageBody`	1		See **Table 5.6**.
`Service`	1…N		Service fragment's root element. from ESG data model.
`id`	1	string	Service ID as specified in id attribute in Service fragment of [2].
`ServiceType`	1	integer	Service type as specified in ServiceType element in Service fragment of [2].
`Name`	1..N	object	Service Name as specified in Name element in Service fragment of [2].
`Description`	0..N	object	Service Description as specified in Description element in Service fragment of [2].
`TargetUserProfile`	0..N	object	Target user profile as specified in TargetUserProfile element in Service fragment of [2].
`Content`	0…N		Content fragment's root element. from ESG data model.
`Programid`	1	string	Content ID as specified in id attribute in Content fragment of [2].
`Name`	1..N	string	Content Name as specified in Name element in Content fragment of [2].
`Description`	0..N	string	Content Description as specified in Description element in Content fragment of [2].

TargetUserProfile	0..N		Target User Profile as specified in TargetUserProfile element in Service fragment of [2].
CARatings	1	string	Content advisory ratings for the content as specified in section 7.3.1 of [3].
Capabilities	1	string	Required capabilities as specified in sa:Capabilities element in Content fragment of [2].
Components	0..N		
componentID	1	string	Component identifier.This string shall be the same as `@componentId` attribute in User Service Bundle Description of MMT [3] or it shall be the string representation of `AdpatationSet@id`, or `ContentComponent@id`, or `Representation@id` value in DASH MPD [3].
componentType	1	unsigned Byte	Type of the continuous component (i.e. audio, video, closed caption). 0 indicates an audio component. 1 indicates a video component. 2 indicates a closed caption component. Values 3 to 255 are reserved.
componentRole	0..1	string	Role or kind of the component.This shall be a string representation of `@componentRole` attribute in User Service Bundle Description of MMT with semantic meaning as defined in [3].
componentName	0..1	string	Human readable name of the component.
ComponentLocation	0..1	String (URI)	URL for access to the component. *Note: PD is not required to stream/ transmit any component to CD.*
FileContentItem	0..N		
FileContentItemLocation	1	String (URI)	URL to access to the file content item.
FileContentItemName	0..1	string	Human readable name of the file content item.
FileContentItemID	1	string	File content item identifier.
FileContentItemType	1	string	File content item's content-type. Obeys the semantics of Content-Type header of HTTP/1.1 RFC 2616 [5].
FileContentItemEncoding	1	string	File content item's content-encoding. Obeys the semantics of Content-Encoding header of HTTP/1.1 RFC 2616 [5].
TimelineInfo			
currentTime	1	date-time	Time location in the content.
Location	0..1	String (URI)	URL for access to the content.

5.6.2.3 Protocol for Current Service Information

HTTP (Request-Response)

5.6.2.4 Message Content for Current Service Information

5.6.2.4.1 CD Request to PD to Receive Current Service Information

A CD may send a HTTP GET request to the PD to request current service information. The request URL and request parameters shall be as follows:

Request URL: `<PD Host URL>/atsc3.csservices.esg.1?<Query>`

URL query parameters `<Query>` shall be as defined in **Table 5.9**.

Table 5.9 Current Service Information Request

Query Parameter	Description
`ServiceInfoType`	This 32 bit field shall indicate type of current service information requested. One or more of following type of service information about the currently running service/ show/ program may be requested: • Request for current show ESG information • Request for current available components for the current show • Request for current available files or non-real-time content for the current show (not including AEA content; see Section 5.7) • Request for current timeline location within the current show The request shall be interpreted as follows based on the value of the query parameter `ServiceInfoType[i]`: • `ServiceInfoType[0]` equal to 1 indicates current show ESG information is requested. `ServiceInfoType[0]` equal to 0 indicates current show ESG information is not requested. • `ServiceInfoType[1]` equal to 1 indicates information about available components for the current show is requested. `ServiceInfoType[1]` equal to 0 indicates information about available components for the current show is not requested. • `ServiceInfoType[2]` equal to 1 indicates information about available files or non-real-time content for the current show is requested. `ServiceInfoType[2]` equal to 0 indicates information about available files or non-real-time content for the current show is not requested. • `ServiceInfoType[3]` equal to 1 indicates information about current timeline location for the current show is requested. `ServiceInfoType[3]` equal to 0 indicates information about current timeline location for the current show is not requested. Bits `ServiceInfoType[4]`- `ServiceInfoType[31]` are reserved for future use.

5.6.2.4.2 PD Current Service Information Response to CD

Upon receiving a request from a CD for the current service information as defined in section 5.6.2.4.1, if the PD supports sending one or more types of information about the current service then it shall include the information in the response as shown below.

The HTTP response `MessageBody` of current service information response shall be JSON formatted and shall conform to JSON schema shown in Annex A, Section A.4. **Table 5.10** describes the structure of the HTTP response in a more illustrative way. The "Description" column in **Table 5.10** gives the semantics of this message's fields.

Table 5.10 Current Service Information Response

FieldName	Description
`PDCDServiceName`	Service name for this service. The "`PDCDServiceName`" property shall be set to a value of "`atsc3.csservices.esg.1`".
`MessageBody`	Message body for the current service information response
`ServiceInfoRespType`	This 32 bit field shall indicate type of current service information returned in the response. One or more of following types of content information about the currently running service/ show/ program may be included in the response. • Current show ESG information

		• Current available components for the current show • Current available files or non-real-time content for the current show (not including AEA content; see Section 5.7) • Current timeline location within the current show The response shall be interpreted as follows based on the value of `ServiceInfoRespType[i]`: • `ServiceInfoRespType[0]` equal to 1 indicates current show ESG information is included in the response. `ServiceInfoRespType[0]` equal to 0 indicates current show ESG information is not included in the response. • `ServiceInfoRespType[1]` equal to 1 indicates information about available components for the current show is included in the response. `ServiceInfoRespType[1]` equal to 0 indicates information about available components for the current show is not included in the response. • `ServiceInfoRespType[2]` equal to 1 indicates information about available files or non-real-time content for the current show is included in the response. `ServiceInfoRespType[2]` equal to 0 indicates information about available files or non-real-time content for the current show is not included in the response. • `ServiceInfoRespType[3]` equal to 1 indicates information about current timeline location for the current show is included in the response. `ServiceInfoRespType[3]` equal to 0 indicates information about current timeline location for the current show is not included in the response. Bits `ServiceInfoRespType[4]`-`ServiceInfoRespType[31]` are reserved for future use.
	`ESGInfo`	Same as 5.6.2.2: Service field and its sub-fields and Program`id`, `Name`, `Description`, `CARatings` properties of Content.
	`Components`	Same as 5.6.2.2 "`Components`" field and its sub-fields.
	`FileContentITem`	Same as 5.6.2.2 "`FileContentItem`" field and its sub-fields.
	`TimelineInfo`	Same as 5.6.2.2 "`TimelineInfo`" field and its sub-fields.

5.6.3 Protocol and Message Content for ESG Communication

5.6.3.1 Protocol

HTTP (Request-Response)

5.6.3.2 Message Content

5.6.3.2.1 CD Request to PD to Receive Partial/Full ESG

A CD may send a HTTP GET request to the PD to request partial or full ESG information. The request URL and request parameters shall be as follows:

Request URL: `<PD Host URL>/ atsc3.csservices.esg.2?<Query>`

The URL query parameters `<Query>` shall be as defined in **Table 5.11**.

Table 5.11 ESG Request

Query Parameter	Description
`ESGRequesttype=0`	Request for ESG information for the current show only. It consists of the Service, Schedule and Content fragments (as defined in [2]) of the current show.
`ESGRequesttype=1`	Request for ESG information for the current service only. It consists of the Service, Schedule and Content fragments (as defined in [2]) of the current virtual channel.
`ESGRequesttype=2`	Request for all ESG information for all available services. It consists of the Service, Schedule and Content fragments (as defined in [2]) of all virtual channels of the ESG that are available to be transferred.

5.6.3.2.2 PD ESG Response to CD

The ESG Response from PD to CD shall be as shown in **Table 5.12**.

The PD ESG Response to CD shall be JSON formatted and the MessageBody shall conform to JSON schema shown in Annex A, Section A.5. **Table 5.12** describes the structure of the ESG response in a more illustrative way. The "Description" column in **Table 5.12** gives the semantics of this message's fields.

Table 5.12 ESG Response

Field Name	Cardinality	Description
`MessageBody`	1	See **Table 5.6**.
`ESGResponseType`	1	`ESGResponseType` equal to 0 indicates that ESG information for only the current show is included in the response. `ESGResponseType` equal to 1 indicates that ESG information for only the current service is included in the response. `ESGResponseType` equal to 2 indicates that all ESG information for all the services is included in the response.
`PDService`	0…N	Container for Service fragment and its sub-elements as defined in [2]. Contains the following element: Service
`PDSchedule`	0…N	Container for Schedule fragment and its sub-elements as defined in [2]. Contains the following element: Schedule
`PDContent`	0…N	Container for Content fragment and its sub-elements as defined in [2]. Contains the following element: Content

When `ESGRequesttype=0` or `ESGRequesttype=1` but the PD is not able to transfer the ESG of the current show segment or virtual channel, the `MessageBody` field shall be transferred with no sub-fields. When `ESGRequesttype=2`, the PD may transfer ESGs of virtual channels having ESGs that are available to be transferred or the PD may respond with a lower value for `ESGResponseType` than requested in the `ESGRequestType` and its associated ESG information.

5.6.4 Protocol and Message Content for Service, Show and Segment Data Communication

5.6.4.1 Protocol

HTTP (Request-Response)

5.6.4.1.1 For Continuous Component

Continuous components shall be accessed via the URL of the `ComponentLocation` field as described in Section 5.6.2.2. CD applications shall use the HTTP GET method to retrieve continuous components via the URL.

5.6.4.1.2 For Adjunct Data (or Files/Data) Component

Adjunct data shall be accessed via the URL of the `FileContentItemLocation` field as described in Section 5.6.2.2. CD applications shall use the HTTP GET method to retrieve adjunct data via the URL. (See Section 5.6.3 for ESG files and Section 5.7 for AEA files.)

5.6.4.2 Message Content

This communication shall be performed via HTTP GET with the URL indicated by the `Location` field described in Section 5.6.2.2.

5.6.5 Protocol and Message Content for Media Timeline Communication

5.6.5.1 Protocol

HTTP (Request-Response) and WebSocket (Notification)

5.6.5.2 Message Content

The media timeline response from PD to CD shall be as shown in **Table 5.13**. The media timeline response message shall be JSON formatted and the `MessageBody` shall conform to JSON schema shown in Annex A, Section A.6. **Table 5.13** describes the structure of the media timeline response in a more illustrative way. The "Description" column in **Table 5.13** gives the semantics of this message's fields.

Table 5.13 Media Timeline Response

FieldName	Cardinality	Description
`MessageBody`	1	See **Table 5.6**.
`absoluteTime`	1	Contains the current UTC time.
`mediaTime`	1	Contains the media time at the current UTC time specified by the `absoluteTime` field.

5.6.6 Protocol and Message Content for Media Playback State Communication

5.6.6.1 Protocol

WebSocket (Notification)

5.6.6.2 Message Content

The message content of the media playback state communication message shall be as specified in Section 5.6.1.3.2.

Fields that are carried in the media playback state information notification message field`MessageBody` from PD to CD shall be as shown in **Table 5.14**.

The media playback state communication message shall be JSON formatted and the `MessageBody` shall conform to JSON schema shown in Annex A, Section A.7. **Table 5.14** describes the structure of the media playback state communication message in a more illustrative way. The "Description" column in **Table 5.14** gives the semantics of this message's fields.

Table 5.14 Media Playback State Information

Field Name	Cardinality	Data type	Description
`MessageBody`	1		See **Table 5.6**.
`MPState`	1	string (enumerated)	Current media playback state for the media ID associated with the media playback state information subscription. The state can be one of the following: "PLAYING", "PAUSED", "STOPPED", "BUFFERING", "UNKNOWN" The "STOPPED" state indicates end of the media stream for the media ID associated with the media playback state information.
`MPSpeed`	0..1	fraction (string)	Current speed of the media state relative to normal speed.

			• Positive MPSpeed values indicate forward playback. Forward playback means media timeline position increases as wall-clock time increases. • Negative MPSpeed values indicate backward playback. Backward playback means media timeline position decreases as wall-clock time increases. • MPSpeed value of 1 indicates forward playback at normal speed. In case of forward playback at normal speed the media timeline increases by the same amount of time as the wall-clock time. • MPSpeed value of –1 indicates backward playback at normal speed. In case of backward playback at normal speed the media timeline decreases by the same amount of time as the wall-clock time. • MPSpeed value of X with X not equal to 0 or 1 indicates playback at X times the normal speed. In case of playback at X times the normal speed the media timeline increases (for positive X values) or decreases (for negative X values) by X times the amount of time as the wall-clock time. • MPSpeed value of 0 is reserved to indicate an UNKNOWN playback speed when the current MPState is "PLAYING". • When MPState is any state other than "PLAYING", MPSpeed shall be equal to value of 0. • When not present MPSpeed is inferred to be equal to 1 when MPState is equal to "PLAYING". • When not present MPSpeed is inferred to be equal to 0 when MPState is equal to any state other than "PLAYING". This field shall be included when the PD supports PVR functionality.
	MediaID	0..1	string	Identifier for the media for which media playback state information subscription is requested. The identifier may uniquely identify the media on the primary device for which the media playback state information subscription is requested. A value of "CURRENT" indicates that the information about the main media currently being played back on PD is requested.

More than one {MPState, MPSpeed, MediaID} could be carried as JSON messages in MessageBody (e.g., for the case of picture-in-picture or multiview use cases).

5.7 Protocol and Message Content for Emergency Alert Messages Communication

5.7.1 Protocol

The following subsections describe the protocols for message content for emergency alert message communication.

5.7.1.1 Introduction

An Advanced Emergency Alert Message (AEAM) may be received by a PD and rendered by the control function of that device. The protocol described in this section describes the transfer of the AEAM to CD on the local area network. This includes a PD application on a PD launching CD applications on CDs and sending the emergency message to these applications for rendering.

The emergency message may be sent to the CDs using two communication means: a WebSocket and a Multicast. The protocol for sending emergency messages to CDs using a WebSocket is described in Section 5.7.1.2 and the protocol for sending emergency messages to CDs using a Multicast Group is described in Section 5.7.1.3. The AEAM Message format is described in Section 5.7.2.

The following functions are distinguished in this mode:

- PD Application: resides in the PD. PD Application is responsible for transferring the emergency message to available CDs in the local area network.
- CDManager: resides in the PD. The CDManager is responsible for discovering CDs with running Launchers and sending Emergency Alert CD application launch information to those Launchers.
- WebSocket Server: resides in the PD. WebSocket Server is responsible for handling WebSocket communications between PD applications and launched Emergency Alert CD applications.
- Multicast Sender: resides in the PD. Multicast Sender is responsible for sending multicast messages from PD applications.
- Launcher: resides in the CD. The Launcher is the application responsible for communicating with the CDManager of the PD and launching the Emergency Alert CD application on the CD.
- Emergency Alert CD application: resides in the CD. Emergency Alert CD application is responsible for receiving the emergency message from the PD on the CD and displaying the result to the user. (See **Figure 5.4**.)

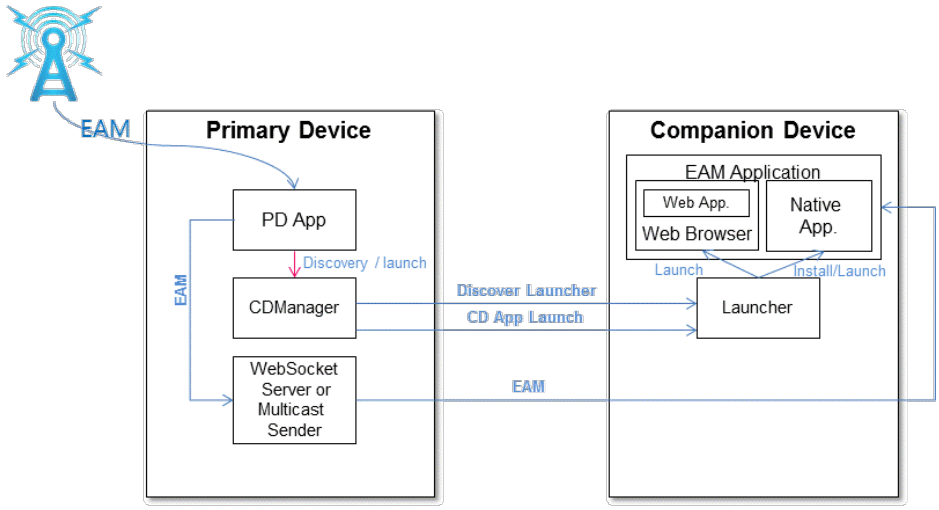

Figure 5.4 Architecture for CD application to PD communication.

The PD, while executing its internal control function, receives an Advanced Emergency Alert Message (AEAM) and in response, the internal control function of a PD that supports this feature shall launch an embedded PD application to render the alert and to manage the process of having the alert rendered on CDs in the local area network.

5.7.1.2 Protocol Using WebSocket Communication

This section describes the Emergency Alert protocol when using a WebSocket as the PD application to CD AEAM application communication path.

As described in Section 5.2.1, the PD application shall issue a `discoverCSLaunchers` method to find all CDs having Launchers available to launch a CD application to receive and render the emergency message.

If no CDs with Launchers are discovered, then the PD application responsible for communication with CDs shall self-terminate.

Otherwise, the PD application shall issue a `getApp2AppLocalBaseURL` method to find the local endpoint of the PD WebSocket communication service.

For each CD with a Launcher, the PD application capable of executing this feature shall launch an Advanced Emergency Alert CD application with a `launchCSApp` method to process the AEAM. Before doing so, the Advanced Emergency Alert CD application to be launched shall be identified by a `LaunchURL`, and the remote port of the WebSocket communication service shall also be identified. Each launched Advanced Emergency Alert CD application shall have access to the URL of the remote endpoint of the PD WebSocket communication service.

The PD application shall then attach to the local endpoint of the PD WebSocket communication service, while the Advanced Emergency Alert CD application shall attach to the remote endpoint of the PD WebSocket communication service.

When communication is established between the PD application and the Advanced Emergency Alert CD application, the PD application shall send the received AEAM to the Advanced Emergency Alert CD application which subsequently processes and renders it.

After the Advanced Emergency Alert message rendering time has expired, the PD application shall terminate WebSocket communication and terminate itself, and the CD applications upon the loss of WebSocket communication shall terminate themselves.

5.7.1.3 Protocol Using Multicast Group Communication

This section describes the Advanced Emergency Alert protocol when using a multicast group as the PD application to Advanced Emergency Alert CD application communication path.

As described in Section 5.2.1, the PD application shall issue a `discoverCSLaunchers` method to find all CDs having Launchers available to launch a CD application to receive and render the emergency message.

For each CD with a Launcher, the PD application shall launch an Advanced Emergency Alert CD application with a `launchCSApp` method especially to process the AEAM. During this step, the PD shall send following information to CD: the Advanced Emergency Alert CD application to be launched shall be identified by a `LaunchURL`, and multicast information consisting of a multicast group address and a multicast port. Each launched Advanced Emergency Alert CD application shall have access to the multicast group address.

A launched Advanced Emergency Alert CD application shall check if it has the multicast group information for emergency alert messages. If the Advanced Emergency Alert CD application does not have the multicast group information for emergency alert messages, it will send a request to the PD at a known end-point. The PD application shall send a response, which includes the multicast information consisting of multicast group address and multicast port. When the CD application has the multicast group information it shall join the multicast group for emergency alert messages using the multicast group address.

After the Advanced Emergency Alert CD applications have been launched, the PD application shall send the AEAM to the multicast group address via the Multicast Sender. When an Advanced Emergency Alert CD application receives the emergency message it shall process and render it.

5.7.2 Message Content

5.7.2.1 PD Notification of Advanced Emergency Alert Message

Fields that are carried in emergency alert messages from PD to CD and their descriptions shall be as shown in **Table 5.15**. The Advanced Emergency Alert communication message shall be JSON formatted and the `MessageBody` shall conform to JSON schema shown in Annex A, Section A.8. The specifications following **Table 5.15** give the semantics of this message's fields.

Table 5.15 Advanced Emergency Alert Message

Field Name	Cardinality	Data Type	Short Description
`MessageBody`	1		See **Table 5.6**.
`AEAT`			Root object
`AEA`	1..N		Advanced Emergency Alert formatted as AEA-MF.
`AEAid`	1	string	The identifier of AEA message.
`issuer`	1	string	The identifier of the broadcast station originating or forwarding the message.
`audience`	1	string	The intended distribution of the AEA message.
`AEAtype`	1	string	The category of the message.
`refAEAid`	0..1	string	The referenced identifier of AEA message. It shall appear when the `AEAtype` is "update" or "cancel".
`priority`	1	integer	The priority of the message
`wakeup`	0..1	boolean	Indication that this AEA is associated with a wake-up event.
`Header`	1	object	The container for the basic alert envelope.
`effective`	1	date-time	The effective time of the alert message.
`expires`	1	date-time	The expiration time of the alert message.
`EventCode`	0..1	Object	
`value`	1	string	A code identifying the event type of the AEA message
`type`	1	string	A national-assigned string designating the domain of the code (e.g. SAME in US, …)
`EventDesc`	0..N	Object	
`value`	1	string	The short plain text description of the emergency event (e.g. "Tornado Warning" or "Tsunami Warning").
`lang`	1	string	The code denoting the language of the respective `EventDesc.value`
`Location`	1..N	object	
`value`	1	string	The geographic code delineating the affected area of the alert message
`type`	1	string	A nationally assigned string designating the domain of the code (e.g. "FIPS" in US, or "SGC" in Canada…)
`AEAtext`	1..N	string	
`value`	1	string	Contains the specific text of the emergency notification
`lang`	1	string	The code denoting the language of the respective field of the alert text
`LiveMedia`	0..1		
`bsid`	1	integer	Identifier of the Broadcast Stream that contains emergency-

			related live A/V service.
serviceId	1	integer	Integer number that identifies the emergency-related A/V Service.
ServiceName	0..N		
name	1	string	A user-friendly name for the service where the LiveMedia is available
lang	1	string	The language of the text described in the name
Media	0..N		Contains the component parts of the multimedia resource.
lang	0..1	string	The code denoting the language of the respective Media
mediaDesc	0..1	string	Text describing the content of the media file
mediaType	0..1	string	Text identifying the intended use of the associated media
uri	1	string	The identifier of the media file
contentType	0..1	string	MIME-Type of media content referenced by Media.uri
contentLength	0..1	unsignedLong	Size in bytes of media content referenced by Media.uri
mediaAssoc	0..1	string	URI of another Media resource with which this attribute is associated

5.7.2.2 Advanced Emergency Alert Message Semantics

The following text specifies the semantics of the properties in the AEAT.

AEAT – Root of the AEAT.

AEA – Advanced Emergency Alerting Message. This is the parent object that has AEAid, issuer, audience, AEAtype, refAEAid, and priority properties plus the following child-objects: Header, AEAtext, and optionally LiveMedia, and Media.

AEA.AEAid – This property shall be a string value uniquely identifying the AEA message, assigned by the station (sender). The AEAid shall not include spaces, commas or restricted characters (< and &). This string shall have the value equal to the value of AEAT.AEA@AEAid attribute of the current Advanced Emergency Alerting Message defined in [3].

AEA.issuer – A string that shall identify the broadcast station originating or forwarding the message. issuer shall include an alphanumeric value, such as call letters, station ID, group name, or other identifying value. This string shall not exceed 32 characters. This string shall have the value equal to the value of AEAT.AEA@issuer attribute of the current Advanced Emergency Alerting Message defined in [3].

AEA.audience – A string that shall identify the intended audience for the message. This value shall be the value of the AEAT.AEA@audience attribute of the current Advanced Emergency Alerting Message defined in [3] .

AEA.refAEAid – A string that shall identify the AEAid of a referenced AEA message. It shall appear when the AEAtype is "update" or "cancel". This string shall have the value equal to the value of AEAT.AEA@refAEAid attribute of the current Advanced Emergency Alerting Message defined in [3].

AEA.AEAtype – A string that shall identify the category of the AEA message. This value shall be the value of the AEAT.AEA@AEAtype attribute of the current Advanced Emergency Alerting Message defined in [3].

AEA.priority – The AEA message shall be the value of the AEAT.AEA@priority attribute of the current Advanced Emergency Alerting Message defined in [3].

`AEA.wakeup` – This property, when present and set to `"true"` shall indicate that the AEA is associated with non-zero `ea_wake_up` bits (See Annex G.2 of [3]). The default value, when not present, shall be `"false"`. This value shall be the value of the `AEAT.AEA@wakeup` attribute of the current Advanced Emergency Alerting Message defined in [3].

`Header` – This object shall contain the relevant envelope information for the alert, including the type of alert (`EventCode`), the time the alert is effective (`effective`), the time it expires (`expires`), and the location of the targeted alert area (`Location`).

`Header.effective` – This date-time shall contain the effective time of the alert message. The date and time shall be represented according to JSON "type": "string", and "format": "date-time". This field shall have value corresponding to the value of the `AEAT.AEA.Header@effective` attribute of the current Advanced Emergency Alerting Message defined in [3].

`Header.expires` – This date/time shall contain the expiration time of the alert message. The date and time shall be represented according to JSON "type": "string", and "format": "date-time". This field shall have value corresponding to the value of the `AEAT.AEA.Header@expires` attribute of the current Advanced Emergency Alerting Message defined in [3].

`EventCode` – An object, which provides information about event code value and type of event.

`EventCode.value` – A string that shall identify the event type of the alert message formatted as a string (which may represent a number) denoting the value itself (e.g., in the U.S., a value of “EVI” would be used to denote an evacuation warning). Values may differ from nation to nation, and may be an alphanumeric code, or may be plain text. Only one `EventCode` shall be present per AEA message. This string shall have the value equal to the value of `AEAT.AEA.Header.EventCode` element of the current Advanced Emergency Alerting Message defined in [3].

`EventCode.type` – This property shall be a nationally assigned string value that shall designate the domain of the `EventCode` (e.g., in the U.S., “SAME” denotes standard FCC Part 11 EAS coding). Values of `type` that are acronyms should be represented in all capital letters without periods.

If `EventCode.type ="SAME"`, then the `EventCode.value` shall be defined as a three letter event code as defined in FCC’s Part 11 rules on EAS (at 47 CFR 11.31(e)).

This string shall have the value equal to the value of `AEAT.AEA.Header.EventCode@type` attribute of the current Advanced Emergency Alerting Message defined in [3].

`EventDesc-` An object, which provides information about event description value and language of event.

`EventDesc.value` – A string that shall contain a short plain text description of the emergency event. This string shall not exceed 64 characters. When the `EventCode` object is present, the `EventDesc` should correspond to the event code indicted in the `EventCode.value` (e.g. an `EventDesc` of “Tornado Warning” corresponds to the EAS `EventCode.value` of `"TOR"`). When an `EventCode` is not present, the `EventDesc` should provide a brief, user-friendly indication of the type of event (e.g., “School Closing”). This string shall have the value equal to the value of `AEAT.AEA.Header.Eventdesc` element of the current Advanced Emergency Alerting Message defined in [3].

`EventDesc.lang`– This property shall identify the language of the respective `EventDesc.value` of the alert message. This property shall be represented by formal natural language identifiers and shall not exceed 35 characters in length as defined by BCP 47 [9]. There shall be no implicit default value. This string shall have the value equal to the value of

AEAT.AEA.Header.Eventdesc@lang attribute of the current Advanced Emergency Alerting Message defined in [3].

Location – An object, which provides information about geographical location value and type of location.

Location.value – A string that shall describe a message target with a geographically-based code. This string shall have the value equal to the value of AEAT.AEA.Header.Location element of the current Advanced Emergency Alerting Message defined in [3].

Location.type – This property shall be string that identifies the domain of the Location code. Note that some primary devices and companion devices may not be capable of determining whether they are located within the signaled location area of the alert. It is suggested that such primary devices and companion devices process the alert as if they were located within the area of the alert.

- If type is equal to "FIPS", then the Location shall be defined as :
  a group of one or more numeric strings separated by commas. Each 6-digit numeric string shall be a concatenation of a county subdivision, state and county codes as defined in FIPS [FIPS] in the manner defined in 47 CFR 11.31 as PSSCCC. Additionally, the code "000000" shall mean all locations within the United States and its territories, and the code “999999” shall mean all locations within the coverage area of the station from which this AEAT originated.
- If type is equal to "SGC", then the Location shall be defined as:
  a group of one or more numeric strings separated by commas. Each numeric string shall be a concatenation of a 2-digit province (PR), a 2-digit census division (CD) and a 3-digit census subdivision (CSD) as defined in SGC. Additionally, the code “00” shall mean all locations within Canada, and the code “9999” shall mean all locations within the coverage area of the station from which this AEAT originated.
- If type is equal to "polygon", then the Location shall define a geospatial space area consisting of a connected sequence of four or more coordinate pairs that form a closed, non-self-intersecting loop.
- If type is equal to "circle", then the Location shall define a circular area represented by a central point given as a coordinate pair followed by a space character and a radius value in kilometers.

Textual values of type are case sensitive, and shall be represented in all capital letters, with the exceptions of “polygon” and “circle”.

This string shall have the value equal to the value of AEAT.AEA.Header.Location@type attribute of the current Advanced Emergency Alerting Message defined in [3].

AEAtext – An object, which provides information about advanced emergency alert message text value and language of the text.

AEAtext.value – A string of the plain text of the emergency message. Each AEAtext object shall include exactly one AEAtext.lang property. For AEAtext of the same alert in multiple languages, shall require the presence of multiple AEAtext objects. This string shall have the value equal to the value of AEAT.AEA.AEAtext element of the current Advanced Emergency Alerting Message defined in [3].

AEAtext.lang– This property shall identify the language of the respective AEAtext field of the alert message. This property shall be represented by formal natural language identifiers as

defined by BCP 47 [9], and shall not exceed 35 characters. There shall be no implicit default value.

`LiveMedia` – An object which provides identification of an A/V service that may be presented to the user as a choice to tune for emergency-related information, e.g., ongoing news coverage. A `LiveMedia` object shall be present if `AEA.wakeup` is `"true"`.

`LiveMedia.bsid` – Identifier of the broadcast stream which contains the emergency-related live A/V service. This field shall have the value equal to the value of `AEAT.AEA.LiveMedia@bsid` attribute of the current Advanced Emergency Alerting Message defined in [3].

`LiveMedia.serviceId` – A 16-bit integer that shall uniquely identify the emergency-related live A/V service. This field shall have the value equal to the value of `AEAT.AEA.LiveMedia@serviceId` attribute of the current Advanced Emergency Alerting Message defined in [3]

`ServiceName` – An object, which provides information about service name and language of the service name.

`ServiceName.name` – A user-friendly name for the service where the live media is available that the receiver can present to the viewer when presenting the option to tune to the `LiveMedia`, e.g., "WXYZ Channel 5." This string shall have the value equal to the value of `AEAT.AEA.LiveMedia.ServiceName` element of the current Advanced Emergency Alerting Message defined in [3]

`ServiceName.lang` – Shall identify the language of the respective `ServiceName.name` property of live media stream. This property shall be represented by formal natural language identifiers and shall not exceed 35 characters, as defined by BCP 47 [9]. There shall be no implicit default value. This string shall have the value equal to the value of `AEAT.AEA.LiveMedia.ServiceName@lang` attribute of the current Advanced Emergency Alerting Message defined in [3]

`Media` – Shall contain the component parts of the multimedia resource, including the `language` (`lang`), description (`mediaDesc`) and location (`uri`) of the resource. Refers to an additional file with supplemental information related to the `AEAtext`; e.g., an image or audio file. Multiple instances may occur within an `AEA` message block.

`Media.lang` – This property shall identify the respective language for each `Media` resource, to help instruct the recipient if different language instances of the same multimedia are being sent. This property shall be represented by formal natural language identifiers as defined by BCP 47 [9], and shall not exceed 35 characters. `Media.lang` shall be present if the `Media.mediaDesc` is present.

`Media.mediaDesc` – A string that shall, in plain text, describe the content of the `Media` resource. The description should indicate the media information. For example, "Evacuation map" or "Doppler radar image" etc. The language of the `Media.mediaDesc` shall be inferred to be same as the language indicated in `Media.lang`. This information may be used by a receiver to present a viewer with a list of media items that the viewer may select for rendering. If this field is not provided, the receiver may present generic text for the item in a viewer UI (e.g., if the `@contentType` indicates the item is a video, the receiver may describe the item as "Video" in a UI list). This string shall have the value equal to the value of `AEAT.AEA.Media@mediaDesc` attribute of the current Advanced Emergency Alerting Message defined in [3].

`Media.mediaType` – This string shall identify the intended use of the associated media. Note that media items identified with this property are typically associated with items that are

automatically handled by the receiver's alert user interface, as opposed to media that is presented in a list to the user for selection. This string shall have the value equal to the value of `AEAT.AEA.Media@mediaType` attribute of the current Advanced Emergency Alerting Message defined in [3].

`Media.uri` – A required property that shall determine the source of multimedia resource files or packages. When a rich media resource is delivered via broadband, this field shall be formed as an absolute URL and reference a file on a remote server. When a rich media resource is delivered via broadcast ROUTE, this field the URL for the resource shall be formed as a relative URL. The relative URL shall match the `Content-Location` attribute of the corresponding File element in the EFDT in the LCT channel [3] delivering the file, or the Entity header of the file.

`Media.contentType` – A string that shall, represent MIME type of media content referenced by `Media.uri`. `Media.contentType` shall obey the semantics of `Content-Type` header of HTTP/1.1 protocol RFC 7231 [8].

`Media.contentLength` – A string that shall, represent size in bytes of media content referenced by `Media.uri` This field shall have the value equal to the value of `AEAT.AEA.Media@contentLength` attribute of the current Advanced Emergency Alerting Message defined in [3].

`Media.mediaAssoc` – An optional property containing a `Media@uri` of another rich media resource with which this media resource is associated. Examples include a closed caption track associated with a video. Construction of `Media.mediaAssoc` shall be as described in `Media.uri` above. This value shall be the value of the `AEAT.AEA.Media@mediaAssoc` attribute of the current Advanced Emergency Alerting Message defined in [3].

5.7.3 Rendering an Advanced Emergency Message

The emergency message may contain text that can be scrolled/displayed in the display of the CD and the emergency message may also contain a URI(s) to rich media content used to support the Advanced Emergency Alert, e.g. a map of the area affected by the alert. When the emergency message is processed by the CD application, the emergency message text may be extracted and then scrolled/ displayed on the CD display screen. The consumer is given the choice of accessing and viewing the rich media content.

5.8 Companion Device APIs

This API enables applications to;

- Discover Companion Devices with a running Launcher Application
- Discover the base URLs of the local and remote endpoints for application to application communication
- Launch or install a CD application on a Companion Device

Connected Launcher Application shall be discovered before launching/installing a CD application by sending the following JSON object to the WebSocket endpoint:

```
{
  “discoverCSLaunchers”
}
```

Description: Reports CD launcher applications on the home network, along with their enumeration ID, a friendly name and their CD OS information.

CD Launcher Applications that are currently connected to the PD shall cause the `onCSDiscovery`.

The protocol for achieving this is out of scope, and not defined by the present document.

The details of what is done during this function call or after this function call depends on the protocol between the PD and the CD launcher application and is implementation specific.

After receiving `discoverCSLaunchers`, the following JSON object shall be returned for each connected CD Launcher Application:

```
{
  “onCSDiscovery” : [
    { true, “enum_id” : Number, “friendly_name” : “String”, “CS_OS_id” : “String”
},
]
}
```

Properties:

`true/false`: returns true to indicate that `discoverCSLaunchers` has completed with no errors, false otherwise.

`enum_id`: An ID for CD Launcher Application. The `enum_id` is expected to be quasi-static. Repeated calls to `discoverCSLaunchers` shall respond with the same `enum_id` unless the CD Launcher Application has been restarted or re-connected. Newly started and connected Launcher Applications on Companion Devices shall generate new `enum_ids`.

`friendly_name`: A CD Launcher Application may provide friendly name, e.g. “Muttleys Tablet”, for a CD application to use. It is optional that this parameter is returned. If it is not returned, it shall be set to empty string “”.

`CS_OS_id`: The CD OS identifier string.

To launch or install a CD application on a Companion Device, the following JSON object shall be sent for the Companion Device identified by the Companion Device enumerations ID (`enum_id`). The action that the CD Launcher Application on the Companion Device will undertake is described by the payload string. The semantics of the payload format shall be as described in clause 14.4.2 of HbbTV [4].

```
{
  “launchCSApp” :  [
    { “enum-id” : Number, “payload” : “String”},
  ]
}
```

Description: Sends a payload string to the CD Launcher application which contains an instruction set for the CD Launcher application to execute. The result of the Launch operation is communicated to the CD application via the `onLaunch`.

Arguments:

`enum_id`: The enumeration ID of the Companion Device to which the payload parameter string shall be sent.

`payload`: See clause 14.4.2 of HbbTV [4] for the definition of format of the payload parameter string.

When the result of the launch operation is known, the WebSocket Server shall return the following JSON object.

```
{
  “onCSLaunch :  [
    { “enum-id” : Number, “error_code” : Number},
  ]
}
```

Properties:

`enum_id`: A unique ID for a CD Launcher Application

`error_code`: See below for the error codes

The error codes given in **Table 5.16** may be carried in the `onCSLaunch` JSON object.

Table 5.16 Error Codes

Error Code	Numeric value	Error Description
`op_rejected`	0	The CD Launcher Application has automatically rejected the operation with no interaction with the user of the Companion Device.
`op_denied`	1	The CD Launcher Application has blocked the operation, but it was blocked by the explicit interaction of the user of the Companion Device.
`op_not_guaranteed`	2	The CD Launcher Application has initiated the instruction (launch or install) without a problem. It is assumed (to the best knowledge of the Launcher Application) that the launch or installation operation has completed successfully.
`invalid_id`	3	The CD Launcher Application that is identified by `enum_id` is no longer available. (i.e., it has become unavailable since discovery occurred).
`general_error`	4	A general error has occurred. The CD Launcher Application knows with certainty that it has failed in its attempt to initiate the instruction (launch or install) received from the CD application.

Since there are certain actions that the Launcher Application may undertake before responding (and thus the PD returns `onCSLaunch` JSON object), there may be a long delay. Applications will therefore be responsible for timing out.

If a CD application needs to use the service endpoints, these parameters should be passed upon launch (as URL query parameters on the application launch URL). The CD application needs to be able to determine the locations of the service endpoints so that it can construct the launch URL before initiating the launch.

The JSON objects below enable a CD application to determine the locations of these service endpoints.

```
{
  "getApp2AppLocalBaseURL"
}
```

Description: Returns the base URL of the application to application communication service local endpoint. The use of this endpoint to communicate between the CD application and the remote client is described in Section 5.5.

The URL retrieved by this method shall end with a slash ('/') character.

```
{
  "getApp2AppRemoteBaseURL"
}
```

Description: Returns the base URL of the application to application communication service remote endpoint.

The URL retrieved by this method shall be the same as the URL carried in the `<X_ATSC_App2AppURL>` elements and shall end with a slash('/') character. See the examples in Section 5.3.1.3.

After receiving `getApp2AppLocalBaseURL` and `getApp2AppRemoteBaseURL`, the following JSON object shall be returned for each connected CD Launcher Application:

```
{
  "ongetApp2AppLocalBaseURL :  [
    { "serviceURL" : "String"},
  ]
}
```

Properties:

`serviceURL`: the base URL of the application to application communication service local endpoint

```
{
  "ongetApp2AppRemoteBaseURL :  [
    { "serviceURL" : "String"},
  ]
}
```

Properties:

`serviceURL`: the base URL of the application to application communication service remote endpoint

Annex A: Schema

A.1 SCHEMA FOR SUBSCRIPTION RELATED MESSAGE STRUCTURE

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Common subscription message structure schema",
    "description": "All subscription related messages use this generic message structure.",
    "ATSCCD_WSMessage": {
    "type": "object",
    "properties": {
    "PDCDMessageVersion":  {"type": "integer","minimum": 0},
    "PDCDservicename": {
        "type": "string",
        "enum": ["atsc3.services.esg.1","atsc3.services.mps.1","atsc3.services.mt.1"]},
    "oneOf":[{"$ref":"#/body/req"},{"$ref":"#/body/cancelreq"},{"$ref":"#/body/resp"},
{"$ref":"#/body/cancelresp"}],
    "body": {
        "req": {
            "properties": {
                "PDCDmessagetype": { "type": "string","enum": ["subscribe", "renew"]},
                "PDCDSubDuration": {"type": "object"}
            },
            "required": ["PDCDmessagetype","PDCDSubDuration"]},
        "cancelreq": {
            "properties": {
                "PDCDmessagetype": {"type": "string","enum": ["cancel"]}
            },
            "required": ["PDCDmessagetype"]},
        "resp": {
            "properties": {
                "PDCDmessagetype": {"type": "string","enum":
["subscribeResponse","renewResponse"]},
                "PDCDSubDuration": {"type": "object"},
                "PDCDRespCode": {"type": "integer"}
            },
            "required": ["PDCDmessagetype","PDCDSubDuration","PDCDRespCode"]},
        "cancelresp": {
            "properties": {
                "PDCDmessagetype": { "type": "string","enum": ["cancelResponse"]},
                "PDCDRespCode": {"type": "integer"}},
            "required": ["PDCDmessagetype","PDCDRespCode"]
        }
       }
     }
   },
   "required": ["PDCDMessageVersion","PDCDservicename"]
}
```

A.2 SCHEMA FOR NOTIFICATION RELATED MESSAGE STRUCTURE

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Notification message structure schema",
    "description": "All notification related messages use this message structure.",
    "ATSCCDMessage": {
    "type": "object",
    "properties": {
       "PDCDMessageVersion":  {"type": "integer","minimum": 0},
       "PDCDservicename": {"type": "string","enum": ["atsc3.services.esg.1",
"atsc3.services.mps.1", "atsc3.services.mt.1"]},
       "MessageBody": {"type": "object"}
                },
   "required": ["PDCDMessageVersion","PDCDservicename"]
}
}
```

A.3 SCHEMA FOR SERVICE AND CONTENT IDENTIFICATION CONTENT

```
{
    "id": "http://atsc.org/version/3.0/cd/ServiceInfo_pd2cd#",
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Service and Content Identification Message Body",
    "description": "Service and Content Identification Message Body from PD to CD uses this
Schema",
    "type": "object",
    "properties": {
        "MessageBody": {
            "type": "object",
            "properties": {
                "Service": {
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {
                            "id": {
                                "type": "string",
                                "format": "uri"
                            },
                            "ServiceType": {
                                "type": "integer",
                                "minimum": 0,
                                "maximum": 255
                            },
                           "Name": {
                                "type": "array",
                                "items": {
                                    "type": "object",
                                    "properties": {
                                        "text": {"type": "string"},
                                        "lang": {"type": "string"}
                                    },
                                    "required": [
                                        "text"
                                    ]
                                },
                                "minItems": 1
                            },
                            "Description": {
                                "type": "array",
                                "items": {
                                    "type": "object",
                                    "properties": {
                                        "text": {"type": "string"},
                                        "lang": {"type": "string"}
                                    },
                                    "required": [
                                        "text"
                                    ]
                                },
                                "minItems": 0
                            },
                            "TargetUserProfile": {
                                "type": "array",
                                "items": {
                                    "type": "object",
                                    "properties": {
                                        "attributeName": {"type": "string"},
                                        "attributeValue": {"type": "string"}
                                    },
                                    "required": [
                                        "attributeName",
                                        "attributeValue"
                                    ]
                                },
                                "minItems": 0
                            }
                        },
                        "required": [
                            "id",
                            "ServiceType",
                            "Name"
                        ]
                    },
                    "minItems": 1
                },
                "Content": {
                    "type": "array",
```

```
"items": {
    "type": "object",
    "properties": {
        "Programid": {
            "type": "string",
            "format": "uri"
        },
        "Name": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "text": {"type": "string"},
                    "lang": {"type": "string"}
                },
                "required": [
                    "text"
                ]
            },
            "minItems": 1
        },
        "Description": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "text": {"type": "string"},
                    "lang": {"type": "string"}
                },
                "required": [
                    "text"
                ]
            },
            "minItems": 0
        },
        "TargetUserProfile": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "attributeName": {"type": "string"},
                    "attributeValue": {"type": "string"}
                },
                "required": [
                    "attributeName",
                    "attributeValue"
                ]
            },
            "minItems": 0
        },
        "CARatings": {"type": "string"},
        "Capabilities": {"type": "string"},
        "Components": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "componentID": {"type": "string"},
                    "componentType": {
                        "type": "integer",
                        "minimum": 0,
                        "maximum": 255
                    },
                    "componentRole": {"type": "string"},
                    "componentName": {"type": "string"},
                    "componentLocation": {
                        "type": "string",
                        "format": "uri"
                    }
                },
                "required": [
                    "componentID",
                    "componentType"
                ]
            },
            "minItems": 1
        },
        "FileContentItem": {
            "type": "array",
            "items": {
                "type": "object",
```

```
                                    "properties": {
                                        "FileContentItemLocation": {
                                            "type": "string",
                                            "format": "uri"
                                        },
                                        "FileContentItemName": {"type": "string"},
                                        "FileContentItemID": {"type": "string"},
                                        "FileContentItemType": {"type": "string"},
                                        "FileContentItemEncoding": {"type": "string"}
                                    },
                                    "required": [
                                        "FileContentItemLocation",
                                        "FileContentItemID",
                                        "FileContentItemType",
                                        "FileContentItemEncoding"
                                    ]
                                },
                                "minItems": 0
                            },
                            "TimelineInfo": {
                                "type": "object",
                                "properties": {"currentTime": {
                                    "type": "string",
                                    "format": "date-time"
                                }},
                                "required": ["currentTime"]
                            },
                            "Location": {
                                "type": "string",
                                "format": "uri"
                            }
                        },
                        "required": [
                            "Programid",
                            "Name",
                            "CARatings",
                            "Capabilities"
                        ]
                    },
                    "minItems": 0
                }
            },
            "required": ["Service"]
        }},
        "required": ["MessageBody"],
        "additionalProperties": false
}
```

A.4 SCHEMA FOR CURRENT SERVICE INFORMATION RESPONSE

```
{
    "id": "http://atsc.org/version/3.0/cd/ServiceInfo_pd2cd#",
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Current Service Message Body",
    "description": "Current Service Message Body from PD to CD uses this Schema",
    "type": "object",
    "properties": {
        "ATSCCS_Message": {
            "type": "object",
            "properties": {
                "PDCDServiceName": {
                    "type": "string",
                    "enum": ["atsc3.csservices.esg.1"]
                },
                "MessageBody": {
                    "type": "object",
                    "properties": {
                        "ServiceInfoRespType": {
                            "type": "integer",
                            "minimum": 0,
                            "maximum": 4294967295
                        },
                        "ESGInfo": {
                            "type": "object",
                            "properties": {
                                "Service": {
                                    "type": "array",
                                    "items": {
                                        "type": "object",
                                        "properties": {
```

```
                "id": {
                    "type": "string",
                    "format": "uri"
                },
                "ServiceType": {
                    "type": "integer",
                    "minimum": 0,
                    "maximum": 255
                },
                "Name": {
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {
                            "text": {"type": "string"},
                            "lang": {"type": "string"}
                        },
                        "required": ["text"]
                    },
                    "minItems": 1
                },
                "Description": {
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {
                            "text": {"type": "string"},
                            "lang": {"type": "string"}
                        },
                        "required": ["text"]
                    },
                    "minItems": 0
                },
                "TargetUserProfile": {
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {
                            "attributeName": {"type": "string"},
                            "attributeValue": {"type": "string"}
                        },
                        "required": [
                            "attributeName",
                            "attributeValue"
                        ]
                    },
                    "minItems": 0
                }
            },
            "required": [
                "id",
                "ServiceType",
                "Name"
            ]
        },
        "minItems": 1
    },
    "Content": {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
                "Programid": {
                    "type": "string",
                    "format": "uri"
                },
                "Name": {
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {
                            "text": {"type": "string"},
                            "lang": {"type": "string"}
                        },
                        "required": ["text"]
                    },
                    "minItems": 1
                },
                "Description": {
                    "type": "array",
                    "items": {
```

```
                                    "type": "object",
                                    "properties": {
                                        "text": {"type": "string"},
                                        "lang": {"type": "string"}
                                    },
                                    "required": ["text"]
                                },
                                "minItems": 0
                            },
                            "CARatings": {"type": "string"}
                        },
                        "required": [
                            "Programid",
                            "Name",
                            "CARatings"
                        ]
                    },
                    "minItems": 0
                }
            },
            "required": ["Service"]
        },
        "Components": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "componentID": {"type": "string"},
                    "componentType": {
                        "type": "integer",
                        "minimum": 0,
                        "maximum": 255
                    },
                    "componentRole": {"type": "string"},
                    "componentName": {"type": "string"},
                    "componentLocation": {
                        "type": "string",
                        "format": "uri"
                    }
                },
                "required": [
                    "componentID",
                    "componentType",
                ]
            },
            "minItems": 1
        },
        "FileContentItem": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "FileContentItemLocation": {
                        "type": "string",
                        "format": "uri"
                    },
                    "FileContentItemName": {"type": "string"},
                    "FileContentItemID": {"type": "string"},
                    "FileContentItemType": {"type": "string"},
                    "FileContentItemEncoding": {"type": "string"}
                },
                "required": [
                    "FileContentItemLocation",
                    "FileContentItemID",
                    "FileContentItemType",
                    "FileContentItemEncoding"
                ]
            },
            "minItems": 0
        },
        "TimelineInfo": {
            "type": "object",
            "properties": {"currentTime": {
                "type": "string",
                "format": "date-time"
            }},
            "required": ["currentTime"]
        }
    },
    "required": ["ServiceInfoRespType"]
}},
```

```
                "required": [
                    "PDCDServiceName",
                    "MessageBody"
                ],
                "additionalProperties": false
            }
    },
    "required": ["ATSCCS_Message"]
}
```

A.5 SCHEMA FOR PD ESG RESPONSE TO CD

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "ESG Response Message Body",
    "description": "ESG Response Message Body from PD to CD uses this Schema",
    "type": "object",
    "properties": {
        "MessageBody": {
            "type": "object",
            "properties": {
                "ESGResponseType": {
                    "type": "integer",
                    "minimum": 0,
                    "maximum": 2
                },
                "PDservice": {
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {"Service": {"type": "object"}}
                    },
                    "minItems": 0
                },
                "PDcontent": {
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {"Content": {"type": "object"}}
                    },
                    "minItems": 0
                },
                "PDschedule": {
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {"Schedule": {"type": "object"}}
                    },
                    "minItems": 0
                }
            },
            "required": ["ESGResponseType"],
            "additionalProperties": false
        }
    },
"required": ["MessageBody"]
}
```

A.6 SCHEMA FOR MEDIA TIMELINE INFORMATION

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Media timeline response message structure schema",
    "description": "Media timeline response related messages use this message structure.",
    "type": "object",
    "properties": {"MessageBody": {
        "type": "object",
        "properties": {
            "absoluteTime": {"type": "string", "format": "date-time"},
            "mediaTime": {"type": "string"}
        },
        "required": ["absoluteTime","mediaTime"]
    }},
"required": ["MessageBody"]
}
```

A.7 SCHEMA FOR MEDIA PLAYBACK STATE INFORMATION

```
{
        "$schema": "http://json-schema.org/draft-04/schema#",
        "title": "Media Playback State Message Body",
        "description": "Media Playback State Message Body from PD to CD use this Schema.",
        "type": "object",
        "properties": {

        "MessageBody": {
            "type": "object",
            "properties": {
              "MPState": {
                "enum": [
                "PLAYING",
                "PAUSED",
                "STOPPED",
                "BUFERRING",
                "UNKNOWN"
                ] },
               "MPSpeed": {
                     "type": "string"
                    },
                "MediaID": {
                     "type": "string"
                }
               },
            "required": ["MPState"],
            "additionalProperties": false },

    },
"required": ["MessageBody"]
}
```

A.8 SCHEMA FOR ADVANCED EMERGENCY ALERT INFORMATION

```
{
    "id": "http://atsc.org/version/3.0/a331/aeat#",
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "ATSC 3.0 Advanced Emergency Alerting Table (AEAT)",
    "description": "AEAT Schema as defined in ATSC 3.0",
    "@context": {"AEAT": "http://www.atsc.org/contexts/3.0/AEAT/V1"},
    "type": "object",
    "properties": {
    "AEAT": {
        "type": "object",
        "properties": {"AEA": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "AEAid": {"type": "string"},
                    "issuer": {"type": "string"},
                    "audience": {"type": "string"},
                    "AEAtype": {"type": "string"},
                    "refAEAid": {"type": "string"},
                    "priority": {"type": "integer"},
                    "wakeup": {"type": "boolean"},
                    "Header": {
                        "type": "object",
                        "properties": {
                            "effective": {
                                "type": "string",
                                "format": "date-time"
                            },
                            "expires": {
                                "type": "string",
                                "format": "date-time"
                            },
                            "EventCode": {
                                "type": "object",
                                "properties": {
                                    "value": {"type": "string"},
                                    "type": {"type": "string"}
                                },
                                "required": [
                                    "value",
                                    "type"
                                ]
                            },
```

```
                "EventDesc": {
                "type": "array",
                 "items": {
                    "type": "object",
                    "properties": {
                        "value": {"type": "string"},
                        "lang": {"type": "string"}
                    },
                    "required": [
                        "value",
                        "lang"
                    ]
                  },
                   "minItems": 0
                },
                "Location": {
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {
                            "value": {"type": "string"},
                            "type": {"type": "string"}
                        },
                        "required": [
                            "value",
                            "type"
                        ]
                    },
                    "minItems": 1
                }
            },
            "required": [
                "effective",
                "expires",
                "Location"
            ]
        },
        "AEAtext": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "value": {"type": "string"},
                    "lang": {"type": "string"}
                },
                "required": [
                    "value",
                    "lang"
                ]
            },
            "minItems": 1
        },
    "LiveMedia": {
            "type": "object",
            "properties": {
                "bsid": {"type": "integer"},
                "serviceId": {"type": "integer"},
                "ServiceName": {
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {
                            "name": {"type": "string"},
                            "lang": {"type": "string"}
                        },
                        "required": [
                            "name",
                            "lang"
                        ]
                    },
                    "minItems": 0
                }
            },
            "required": [
                "bsid",
                "serviceId"
            ]
        },
        "Media": {
            "type": "array",
            "items": {
```

```
                            "type": "object",
                            "properties": {
                                "lang": {"type": "string"},
                                "mediaDesc": {"type": "string"},
                                "mediaType": {"type": "string"},
                                "uri": {
                                    "type": "string",
                                    "format": "uri"
                                },
                                "contentType": {"type": "string"},
                                "contentLength": {"type": "integer"},
                                "mediaAssoc": {"type": "string", "format": "uri"}
                            },
                            "required": ["uri"]
                        },
                        "minItems": 0
                    }

                },
                "required": [
                    "AEAid",
                    "issuer",
                    "audience",
                    "AEAtype",
                    "Header",
                    "AEAtext"
                ]
            },
            "minItems": 1,
            "maxItems": 1
        }},
        "required": ["AEA"],
        "additionalProperties": true
       }
    },
    "required": ["AEAT"]
}
```

Annex B: Usage Scenarios

Several usage scenarios are described below.

Scenario A: Julio is watching a broadcast concert of his preferred rock & roll band on the TV screen (primary device). A notification pop-up on the TV informs him that alternative camera views of the concert presenting each musician are available through a dedicated application on his CD. Julio launches that application which informs Julio that close-ups of the guitarist, bassist, singer and drummer are available. Julio selects the guitarist during the guitar solo and switches to the drummer later in the song. Media content on the TV screen and the companion screen are synchronously rendered.

Scenario B: For a program being watched on TV (primary device) Mary is interested in hearing video description for the visually impaired, but does not wish to enable that for all the viewers in the room. Using an app on her CD she discovers the various audio tracks available and selects the description track for playing on her CD. John is hearing impaired and wants to read closed captions with sound description. Using an app in his CD, he discovers the various options for closed captions and selects the one with audio description to display on his CD. Hector prefers voice over-dubs instead of reading Spanish subtitles. He has a CD app that has a text-to-voice function. Using his CD he discovers the Spanish subtitles and uses his app to convert the text to voice which he listens to via his headphones.

Scenario C: Jane is watching her favorite game show on TV (primary device). A notification pop-up on the TV informs her that she can play along on her tablet through a dedicated tablet app. She launches that app on her tablet and she is able to play along with the game show in real time. Each question is presented to her on her tablet at the same time as in the show, and her response times are limited to the response time the contestants on the show have. Her score is tracked by the app and she can also see her ranking among other viewers who are also playing along using the tablet app.

Scenario D: George launches an On Demand app on his main TV (primary device). The TV app requests some demographic information from George so that it can make program recommendations for George. The TV app suggests a companion tablet app that George can download to make data entry easier. George downloads and launches the tablet app. The tablet app offers George the data entry fields. George completes the data entry on his tablet and the information is registered in the TV app. The TV app recommends several On Demand programs to George based on his entries. George uses his tablet to select one of the recommended programs to be presented on his TV.

Alternatively, George uses his tablet to select one of the recommended programs to be presented on his tablet instead of the main TV.

Scenario E: Laura is watching her favorite program in the living room on her TV (primary device). She has a variety of things she needs to do around the house but does not want to miss any of her show. She launches an app on her tablet (CD) that allows her to watch her show on her tablet as well as on her TV. She continues watching her show on her tablet as she moves from room to room.

While Laura is in the laundry room, an emergency alert message is broadcast. The message appears on her tablet. The tablet also informs her that there is a video of the event that she can view if she wishes. She selects the video and begins to watch. She follows the instructions that the emergency message conveys.

End of Document