ATSC Standard: Companion Device

Doc. A/338:2019
2 October 2019

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 and recommended practices for digital television. ATSC member organizations represent the broadcast, broadcast equipment, motion picture, consumer electronics, computer, cable, satellite, and semiconductor industries. ATSC also develops digital television implementation strategies and supports educational activities on ATSC standards. ATSC was formed in 1983 by the member organizations of the Joint Committee on Inter-society 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). For more information visit www.atsc.org.

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
A/338:2017 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
Candidate Standard revision of A/338:2017 approved	2 January 2019
CS update approved	11 July 2019
A/338:2019 Standard approved	2 October 2019

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 (PD) and an ATSC 3.0 Companion Device (CD). 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 as 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 A.

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 – 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 Revision: Signaling, Delivery, Synchronization and Error Protection," Doc. A/331:2019, Advanced Television Systems Committee, Washington, D.C., 3 May 2019.

[3] ATSC: "ATSC Standard: ATSC 3.0 Interactive Content," Doc. A/344:2019, Advanced Television Systems Committee, 2 May 2019.

[4] ATSC: "ATSC Standard: ATSC 3.0 Security and Service Protection," Doc. A/360:2019, Advanced Television Systems Committee, 20 August 2019.
[5] ETSI: "Hybrid Broadcast Broadband TV," Document ETSI TS 102 796 v1.4.1, European Telecommunications Standards Institute, European Broadcasting Union, August 2016.
[6] IETF: "The WebSocket Protocol," RFC 6455, Internet Engineering Task Force, December 2011. https://tools.ietf.org/html/rfc6455
[7] IETF: "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content," RFC 7231, Internet Engineering Task Force, June 2014. https://tools.ietf.org/html/rfc7231
[8] IETF: "Tags for Identifying Languages," BCP 47, Internet Engineering Task Force, Reston, VA, September 2009. https://tools.ietf.org/html/bcp47

2.2 Informative References

There are no Informative References in the current version of this Standard.

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 Abbreviations

The following acronyms and abbreviations are used within this document.

AEA – Advanced Emergency Alert
AEAT – Advanced Emergency Alert Table
API – Application Programming Interface
ATSC – Advanced Television Systems Committee
BA – Broadcaster Application (PD Application)
CD – ATSC 3.0 Companion Device
EAM – Emergency Alert Message
ESG – Electronic Service Guide
HbbTV – Hybrid Broadcast Broadband Television
HTML5 – Hyper-Text Markup Language 5
HTTP – Hyper-Text Transfer Protocol
ID – Identifier
IEEE – Institute of Electrical and Electronic Engineers
IETF – Internet Engineering Task Force
JSON – JavaScript Object Notation
NRT – Non-Real Time
OS – Operating System
PD – ATSC 3.0 Primary Device
RFC – Request for Comments
SSDP – Simple Service Discovery Protocol
ST – Search Target
TV – TeleVision
URI – Uniform Resource Identifier
URL – Uniform Resource Locator
USN – Unique Service Name

3.4 Terms

The following terms are used within this document.

Broadcaster Application – See A/344 [3] for a detailed definition. The Broadcaster Application operates in the User Agent component of the Primary Device (PD) architecture.

CD Application – A software module operating on a Companion Device that interacts with the Primary Device and, perhaps, with the Broadcaster Application operating on the PD.

CD Launcher – A software application or module operating on a Companion Device that is responsible for installing and/or launching CD Applications.

CD Manager – A module on the Primary Device that manages communications with the CD Launcher. It maintains a list of discovered Companion Devices and allows the PD-based Broadcaster Application to install, launch and communicate with CD Applications.

Companion Device – 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.

Primary Device – The Primary Device is the primary receiver and is used to present the primary content. The use of Primary Device and PD in this standard is equivalent to the use of Receiver and Reference Receiver Model in A/344 [3].

3.5 Extensibility

The JSON-RPC 2.0 Specification provides extensibility based on `jsonrpc` property (See Annex E of A/344 [3]). In addition, the objects described in this standard are defined in JSON which is a highly extensible format. Components using this protocol are expected to ignore any properties or fields that are not understood and to only process attributes and values that are known.

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 Application, PD to CD Application communication and CD Application to PD communication. Cross-Origin requests as described in HbbTV Clause 14.8 [5] 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.

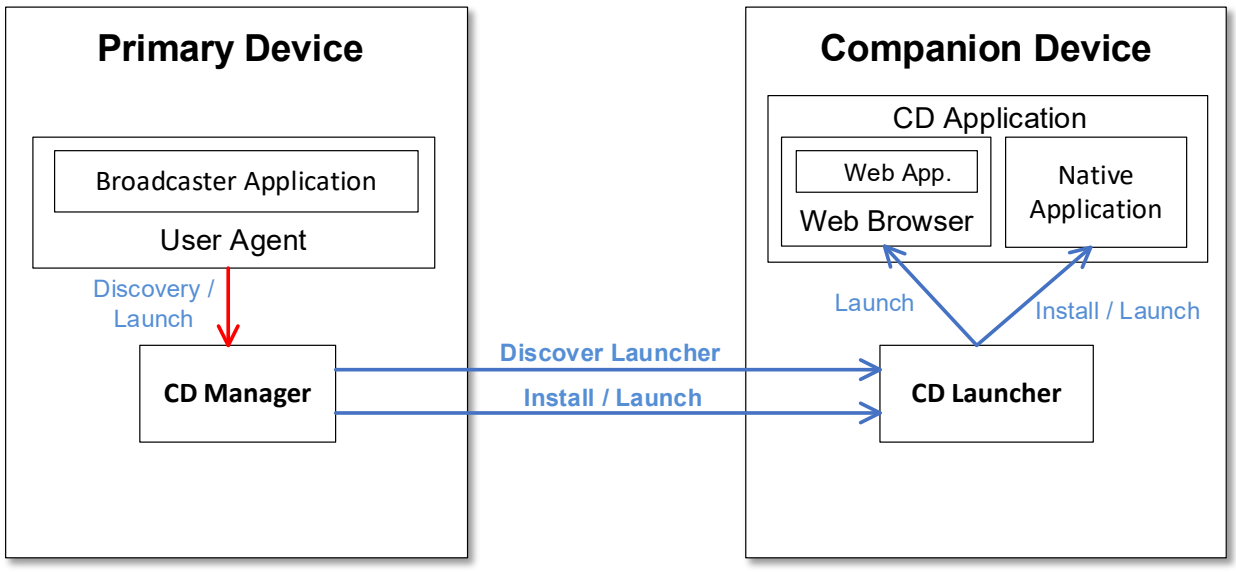

Figure 5.1 Architecture for launching a companion device application.

The following functions are distinguished in this architecture:

- User Agent: running the Broadcaster Application, consists of using HTML5 and associated web technologies. Both the Broadcaster Application and the interactive application environment are described in A/344, Interactive Content [3].
- CD Manager: resides in the PD. The CD Manager is responsible for discovering the CDs with running CD Launchers and sending application launch/install information to those CD Launchers. The Broadcaster Application shall manage the CD Manager through the protocol described in Section 5.4.
- CD Launcher: resides in the CD. The CD Launcher is responsible for communicating with the CD Manager of the PD and launching and/or installing the CD application. The requirements on the CD Launcher shall be as described in HbbTV Clause 14.3.2 [5].

To launch a CD application, the Launch CD Application API described in Section 5.8.2 shall be used.

The protocol for launching a CD application from a CD Manager is as follows. The CD Manager requests the launch of the CD application by sending an HTTP POST request to the Application-URL of the CD Launcher. The Application-URL of the CD Launcher is obtained from using the discovery in Section 5.3. The BODY data of the HTTP POST request shall contain the value of the `parameters` property in the Launch CD Application API request, indicating the CD application to be launched (see Section 5.8.2).

5.2.2 Application to Application Communication

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

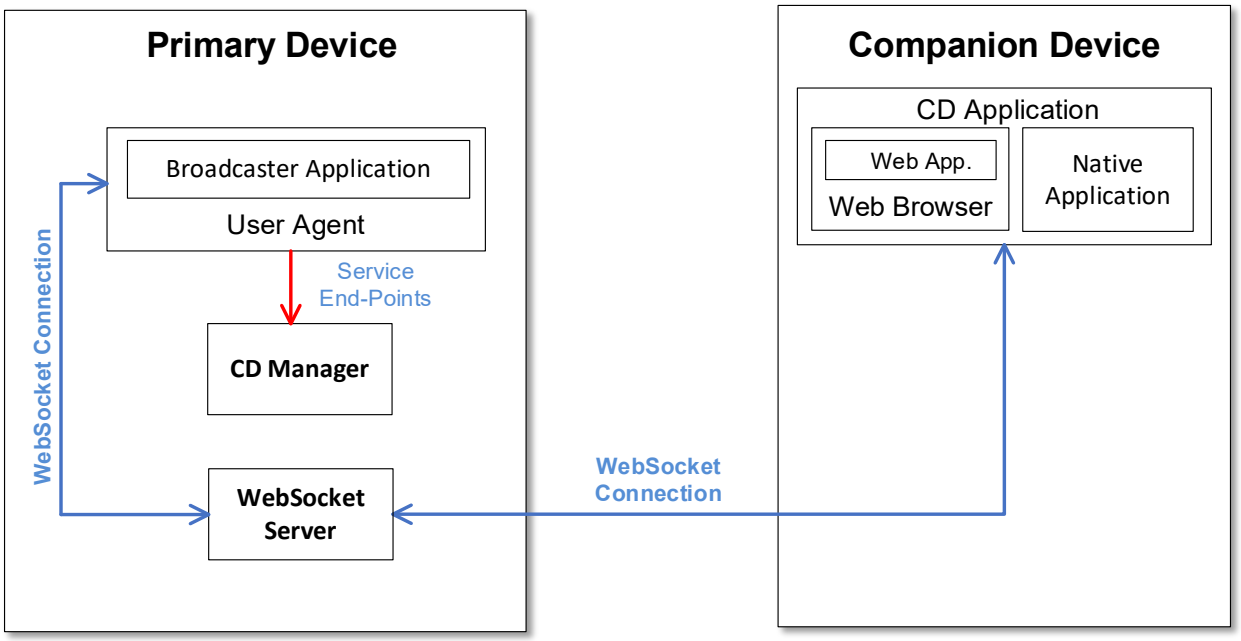

Figure 5.2 Architecture for application-to-application communication.

The following functions are distinguished in this architecture:

- CD Manager: responsible for providing service endpoints for application-to-application communication. The Query Companion Devices API shall be to obtain service endpoints for the communication and is defined in Section 5.8.1.

- WebSocket Server: resides in the PD. The WebSocket Server is responsible for handling web socket connections to/from PD applications and to/from CD applications. The WebSocket protocol is defined in [6] and the 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 HbbTV Clause 14.5 [5] with the exception that with respect to Section 14.5.4, the "wss:" scheme is supported in addition to the "ws:" scheme.

Applications should not use both "ws:" and "wss:" schemes 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 Launch CD Application API parameters (see Section 5.8.2). 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.

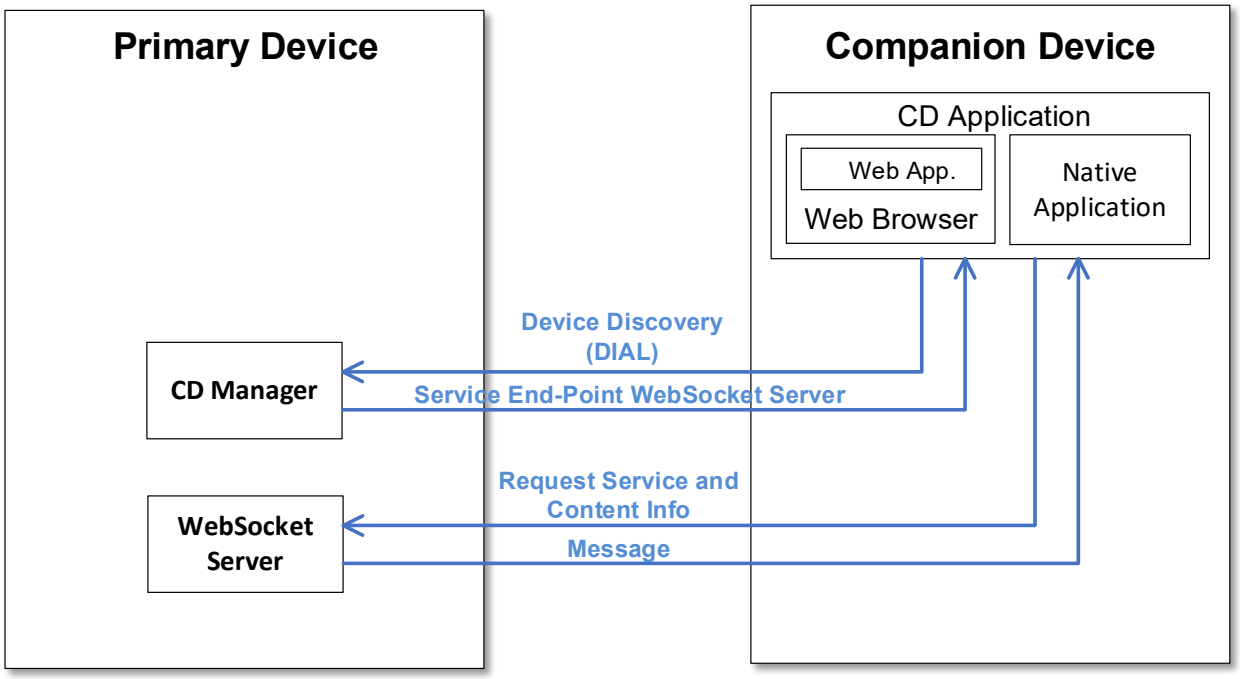

Figure 5.3 Architecture for CD application to PD communication.

- CD Manager: resides in the PD. The CD Manager shall be responsible for responding to discovery requests from CD Applications and for providing the service endpoints of its WebSocket Server.
- 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 WebSocket protocols.

A CD Application may establish communication with the server providing services on a PD. In doing so, the CD Application shall first discover the PD and in the process obtain the remote endpoint of its WebSocket Server. At this point, the CD Application may obtain service and content information through the WebSocket Server. The CD Application shall first establish a WebSocket connection and then send a request for service and content information through this connection. If an encrypted connection is established between a CD application and PD for information exchange, then methods defined in A/360 Section 5.6 [4] shall be used.

5.3 Protocol for Discovery

Both the PD and the CD Application can send 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, perhaps 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 Application Discovery of PDs

The mechanism for CD Applications to discover PDs shall be performed as described in HbbTV Clause 14.7 [5] and further modified as described below.

5.3.1.1 Introduction

In the situation where a CD Application has been launched by a Broadcaster Application, information regarding the location of the service endpoints exposed by the PD may be conveyed as parameters in the Launch CD Application API as described in Section 5.8.2.

However, to accomplish the CD Application to PD communication described in Section 5.2.3, the CD Application needs to be able to discover the location of the WebSocket Server endpoint 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, HbbTV Clause 14.7.2 [5] shall apply only.

The WebSocket Server endpoint URL for CD Application to PD communication, as given in Section 5.2.3, shall be the Application Resource URL for HbbTV [5] 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 HbbTV Clause 14.7 [5]:

```
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://cd.services.broadcaster.com/
```

Device Description Response:
The PD responds with an `HTTP/1.1 OK` header containing the `Application-URL` as described in HbbTV Clause 14.7 [5]:

```
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://cd.services.broadcaster.com/4
```

Application Information Response:
An HTTP response is returned as follows

Header:

```
HTTP/1.1 200 OK
Origin: http://cd.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_WSURL>URL of WebSocket comm. endpoint
           </X_ATSC_WSURL>
        <X_ATSC_UserAgent>Value of ATSC UA header
           </X_ATSC_UserAgent>
    </additionalData>
</service>
```

<X_ATSC_App2AppURL> – supplies the Application to Application WebSocket endpoint of the PD.

<X_ATSC_WSURL> – supplies 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 the 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 the CD 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

A CD Application may be launched by a PD Broadcaster Application using the Launch CD Application API described in Section 5.8.2. If the Broadcaster Application intends to use application-to-application communication, the parameters of the Launch CD Application request shall supply the remote endpoint of a WebSocket service. The endpoints for both connections may be obtained using the Query Companion Devices API setting the `includeWsEndpoints` parameter to '`true`' (see Section 5.8.1). The CD Application may use the endpoint to communicate with the PD Broadcaster Application.

In addition, the CD Application may elect to be automatically launched when a notification occurs on the PD for which the CD Application is registered. Details of this feature are described in Section 5.6.1.

5.5 Application to Application Communication

Application-to-application communication shall be through a WebSocket Server on the PD. Both the "wss:" and "ws:" schemes are allowed though secure WebSocket connections are preferred. The local and remote endpoints of the WebSocket server for both the Broadcaster Application and the CD Application are provided in the Query Companion Devices API (see Section 5.8.1). The remote WebSocket service endpoint is supplied to the CD Application through the Launch CD Application API (see Section 5.8.2).

5.6 Companion Device Application to Primary Device Communication

CD application to PD communication shall consist of the following:

- 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 or receives notifications via the WebSocket service endpoint.

- A PD responds to CD requests or provides notifications via the WebSocket connection.

There are two service endpoints. The HTTP service endpoint is used to discover the WebSocket service endpoint. The WebSocket service endpoint provides all other communication between the PD and CD. In addition, if a Broadcaster Application is running on the PD, a second WebSocket service endpoint will be provided to allow application to application communication (see Section 5.5). 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 [4] shall be used.

The WebSocket communication between the CD and PD shall conform to JSON-RPC 2.0 as described in Section 8.3 of A/344 [3]. The JSON-RPC 2.0 specification is duplicated in an annex of A/344.

Some of the APIs provided to Broadcaster Applications operating under the auspices of the PD have direct applicability to CD Applications. Table 5.1 provides the list of APIs that shall be provided through the WebSocket interface supplied to a CD.

Table 5.1 Applicable APIs

WebSocket API	A/344 API Section References
Notification Subscribe / Unsubscribe	9.7.6
Current Service Query and Change Notification	9.2.3, 9.3.3
Service Guide Query and Change Notification	9.2.10, 9.3.10
Advanced Emergency Alert Query and Change Notification	9.2.9, 9.3.9
Receiver Media Playback Query and Change Notification	9.14

To receive notifications from the PD, the CD must use the Notification Subscription API described in A/344 Section 9.7.6.1 [3]. The CD may use any of the `msgTypes` defined in A/344 Table 9.3. The PD shall support the subscriptions to the Notifications defined in Table 5.1 but may also provide additional notification support. The notifications supported are enumerated in the response to the subscription request.

5.6.1 Automatic Notification Launch

A CD Application may register to be launched automatically when a notification occurs. When a notification occurs, the PD shall first determine if the CD Application is currently connected to the WebSocket Server CD Application to PD communication endpoint. If not, the CD Manager within the PD shall communicate with the CD Launcher registered for the particular CD to launch the appropriate CD Application. Once the CD Application has connected to the WebSocket Server endpoint, the notification can then be sent.

To register for automatic launch, the CD Application shall include a `launchParams` object with the subscription request. This property extends the Integrated Subscribe schema described in A/344 Section 9.7.6.1 [3] as follows:

params JSON Schema:

```
{
   "type": "object",
   "properties": {
      "msgType": {
         "type": "array",
         "items": {
            "type": "string",
            "enum": [List of msgTypes Column in A/344 Table 9.3]
         }
      },
      "launchParams": {"type": "string"}
   },
   "required": ["msgType"]
}
```

`launchParams` – This optional string shall be as described in HbbTV Clause 14.4.2 [5]. Inclusion of this property in the subscription request indicates that if the CD Application making the request is not connected to the PD WebSocket Server endpoint at the time of the requested notification, the PD shall use the `launchParams` to launch the CD Application, waiting for the CD Application to connect before sending the notification.

The PD shall respond as described in A/344 Section 9.7.6.1 but may also provide an error code, `'-20' 'Automatic Launch Not Supported'`, if it cannot honor the automatic launch request. If an error occurs, the subscription is not successful and must be resubmitted. If automatic launch is not supported, the CD Application may resubmit the request without the `launchParams` to receive notifications. A PD may also ignore the `launchParams` object and accept the notification subscription without error. In that case, no automatic launch is performed by the PD and the notification is only sent if the CD Application remains connected.

If a CD Application disconnects from the WebSocket Server and has not supplied a `launchParams` object, then the PD shall consider the CD Application to be disconnected and automatically unsubscribe the CD Application from any notifications it has subscribed to.

For example, the CD Application wishes to subscribe to the "`AEAT`" notification to receive advanced emergency alerts and to be automatically launched when a new notification is received if not already operating. The following request would be sent to the PD:

```
--> {
    "jsonrpc": "2.0",
    "method": "org.atsc.subscribe",
    "params": {
        "msgType": ["AEAT"],
        "launchParams": {
            "launch": [
               {"launchUrl": "https://www.xyza-apps.com/alerting.html",
                "appType": "native"}
            ]
        }
    }
    "id": 51
}
```

Upon success, the PD might respond:

```
<-- {
    "jsonrpc": "2.0",
    "result": {
        "msgType": ["AEAT"],
    }
    "id": 51
}
```

5.7 Emergency Alert Communication

An Advanced Emergency Alert Table (AEAT) may be received by a PD and rendered by the control function of that device. The protocol described in this section supports the transfer of the AEAT to a CD on the local area network. This includes a PD CD Manager launching CD Applications and sending the emergency message to those CD Applications for rendering.

The AEAT is communicated to the CDs using two mechanisms: a WebSocket notification or a WebSocket query as described in Section 5.6. The AEAT schema is described in Section 6.5 of A/331 [2].

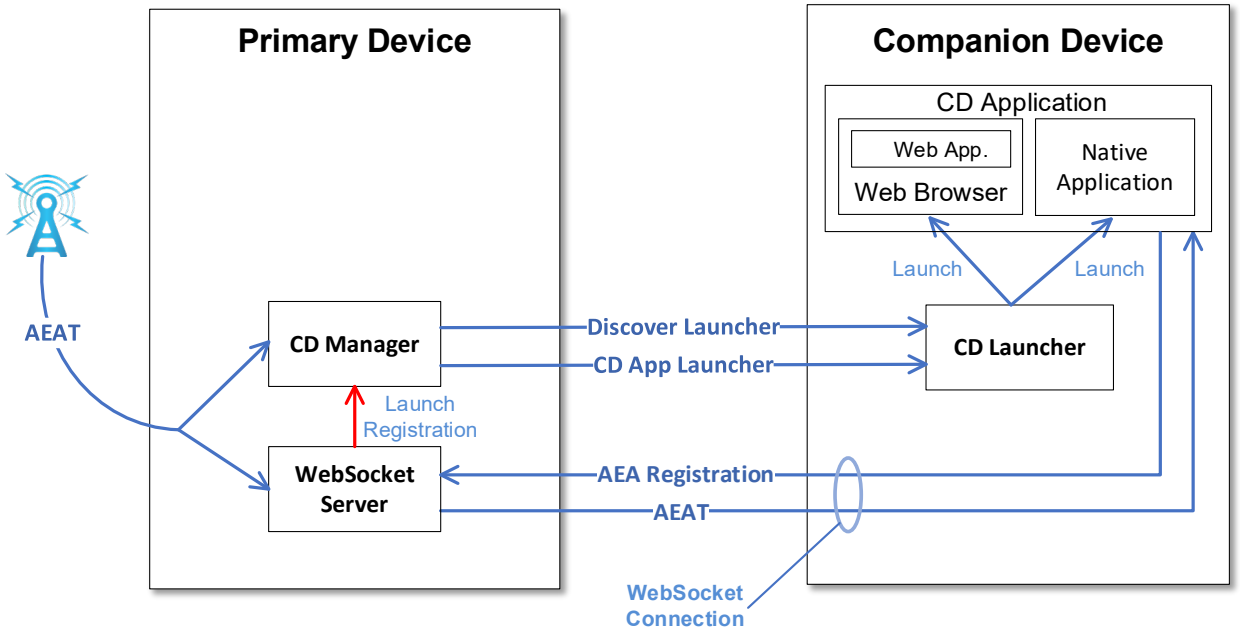

Figure 5.4 Architecture for AEAT Communication to CD with Launch.

The following functions are distinguished in this mode (see Figure 5.4):

- CD Manager: resides in the PD. The CD Manager is responsible for discovering CDs with running CD Launchers and sending CD Application launch information to those CD Launchers. The CD Application registers with the PD as part of the AEAT notification subscription request if the CD Application should be launched when the PD receives a new AEAT.

- WebSocket Server: resides in the PD. The WebSocket Server is responsible for handling WebSocket communications between CD Applications and various functions within the PD.
- CD Launcher: resides in the CD. The CD Launcher is responsible for communicating with the CD Manager of the PD and launching the CD Application on the CD when requested.
- CD Application: resides in the CD. The CD Application is responsible for receiving the AEAT notification from the PD and displaying the result to the user.
- There are two modes of operation for handling Advanced Emergency Alert events depending on whether a Broadcaster Application is involved or not. These are described in the following subsections.

5.7.1 Broadcaster Application AEA Support

The AEA support mechanism when a Broadcaster Application is involved is similar to any other application-to-application communication between the PD and CD. The Broadcaster Application can subscribe to the Alerting Notifications to get AEAT messages. Subscribing to AEAT alerts indicates to the PD that the Broadcaster Application wishes to process the incoming emergency alerts.

The Broadcaster Application can then use the Query Companion Devices API to determine if any CDs are available to launch CD Applications on. If so, the Broadcaster Application can use the Launch CD Application API to install or start the CD Application and initiate an application-to-application communication path. Depending on the facilities within each application, the Broadcaster Application can forward the AEAT directly or provide some derived data structure through the application-to-application WebSocket connection. The details of this communication are out of scope of this specification.

Alternatively, once launched, the CD Application can subscribe to AEAT Notifications directly and receive the most recent AEAT. In this case, the Broadcaster Application would simply operate as a launch mechanism for the CD Application interested in AEATs. Note that this capability may be necessary for PDs that support Broadcaster Applications but do not support automatic launch.

5.7.2 Direct PD AEA Support

In the direct PD AEA support scenario, the PD, while executing its internal control function, receives an Advanced Emergency Alert Table (AEAT) and in response, the internal control function of a PD that supports this feature shall communicate with the CD Manager to manage the process of having the alert rendered on CDs in the local area network.

The CD Manager will determine the set of CD Applications that have registered to be launched when an AEAT notification is received. CD Applications register for automatic notification launch by supplying a `launchParams` object when subscribing to the notification as described in Section 5.6.1. Once each CD Application has launched and connected to the CD Application to PD WebSocket interface, the CD Manager shall send the AEAT notification.

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 AEAT is processed by the CD Application, the emergency message text may be extracted and then scrolled or displayed on the CD display screen. The consumer is given the choice of accessing and viewing

the rich media content. Note that the PD shall make the relative NRT rich media content available to the CD Application through the root Web Server address supplied at discovery (see Section 5.3).

5.8 Companion Device APIs

This APIs described in this section enable a Broadcaster Applications to perform the following actions:

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

The following subsections extend the A/344 API collection, providing an interface to manage CD applications. It extends the supported methods described in Section 9 of A/344 which, in turn, are based on the Web Socket interface protocol described in Section 8 of that same standard [3].

5.8.1 Query Companion Devices API

The Broadcaster Application may discover the available CD Launchers and, optionally, the communication WebSocket service endpoints by using the Query Companion Devices interface. The PD will respond with a list of available CD Launchers along with the WebSocket service endpoints for each CD, if requested. The protocol for registering CD Launchers is out of scope, and not defined by the present document. The details of what is done during this method call depends on the protocol between the PD and the CD Launcher and is implementation specific.

The Query Companion Devices API shall be defined as follows:

method: "`org.atsc.query.companionDevices`"

params: A Boolean value selecting whether the WebSocket service endpoints will be returned or not. A value of 'true' requests the endpoints be included while the default value of 'false' indicates that the PD should not include the endpoints. The default is taken if no parameters are provided. Not requesting the WebSocket service endpoints indicates that the Broadcaster Application has no intention of using application-to-application communication and the PD can avoid allocating resources to this activity.

params JSON Schema:

```
{
  "type": "object",
  "properties": {
    "includeWsEndpoints": {"type": "boolean"}
  }
}
```

`includeWsEndpoints` – This optional parameter indicates, if 'true', that the response shall include the two WebSocket service endpoint URLs for each CD. If this parameter is not provided or is set to 'false', the WebSocket service endpoint URLs shall not be provided in the response.

Response:

result: A JSON object containing a list of CD Launchers, along with their enumeration ID, a human-friendly name and their CD OS information.

result JSON Schema:

```
{
   "type": "object",
   "properties": {
      "launchers": {
         "type": "array",
         "items": {
            "type": "object",
            "properties": {
               "id": {"type": "integer"},
               "name": {"type": "string"},
               "cdOs": {"type": "string"},
               "localEndpointUrl": {"type": "string", "format": "uri"},
               "remoteEndpointUrl": {"type": "string", "format": "uri"},
               "required": ["id", "cdOs"]
            }
         }
      },
      "required": ["launchers"]
   }
}
```

`launchers` – An array of the available CD Launchers known to the CD Manager. Note that this list may be empty if no CD devices have been detected.

`id` – A required, unique identifier for the CD Launcher. The `id` is expected to remain constant as long as the CD Launcher remains connected. Repeated calls to the Query CD Launchers API shall respond with the same `id` unless the CD Launcher has been restarted or re-connected. Newly started and connected CD Launchers shall generate new `id`s.

`name` – A CD Launcher may provide a human-friendly name, e.g. “Muttleys Tablet”, for a CD application to use. This parameter is optional.

`cdOs` – A required identifier string for the operating system of the CD.

`localEndpointUrl` – Defines the local WebSocket service endpoint URL that the Broadcaster Application may use to communicate with the associated CD Application. The value of the URL shall end with a slash (‘/’) character. This parameter is provided only if the request includes the `includeWsEndpoints` parameter with a value of ‘true’.

`remoteEndpointUrl` – Defines the remote WebSocket service endpoint URL that the CD Application may use to communicate with the associated Broadcaster Application. The URL retrieved by this method shall be the same as the URL carried in the `<X_ATSC_App2AppURL>` element supplied in the CD Application launch parameters and shall end with a slash (‘/’) character. See the examples in Section 5.3.1.3. This parameter is provided only if the request includes the `includeWsEndpoints` parameter with a value of ‘true’.

For example, the Broadcaster Application wishing to determine if any Companion Devices are registered with the receiver would issue the following request:

```
--> {
    "jsonrpc": "2.0",
    "method": "org.atsc.query.companionDevices",
    "params": {
        "includeWsEndpoints": "true"
    }
    "id": 501
}
```

The PD might respond with:

```
<-- {
    "jsonrpc": "2.0",
    "result": {
        "launchers": [
          { "id": "1",
            "name": "XYZA: TV Application",
            "cdOs": "Android",
            "localEndpointUrl": "wss://127.0.0.1/cd1app2app/",
            "remoteEndpointUrl": "wss://192.168.10.21/pd/cdApp1"},
          { "id": "2",
            "name": "XAZX: NextGen TV Application",
            "cdOs": "iOS XII",
            "localEndpointUrl": "wss://127.0.0.1/cd2app2app/",
            "remoteEndpointUrl": "wss://192.168.10.21/pd/cdApp2" }
        ]
    },
    "id": 501
}
```

5.8.2 Launch CD Application API

The Launch CD Application API shall be used to launch or install a CD Application on a Companion Device. The action to be taken by the CD Launcher on the Companion Device is provided by the launch `parameters` string.

The Launch CD Application API shall be defined as follows:

method: "`org.atsc.cdApp.launch`"

params: The ID of the CD Launcher to be used to launch the CD Application and the launch parameters communicated to the CD Launcher.

params JSON Schema:

```
{
   "type": "object",
   "properties": {
      "id": {"type": "integer"},
      "parameters": {"type": "string"},
      "required": ["id", "parameters"]
   }
}
```

`id` – A required ID for the CD Launcher to be used to launch the CD Application. The `id` is obtained using the Query CD Launchers API defined in Section 5.8.1.

`parameters` – This required string shall be as described in HbbTV Clause 14.4.2 [5].

Response:

result: A null object upon success.

error: On error, the PD shall return an error object containing either an error code as described in A/344 Section 8.3 [3] or one of the error codes defined in Table 5.2 below.

Table 5.2 Error Codes

Numeric value	Error Description
-14	The CD Launcher has automatically rejected the operation with no interaction with the user of the Companion Device.
-15	The CD Launcher has blocked the operation, but it was blocked by the explicit interaction of the user of the Companion Device.
-16	The CD Launcher has initiated the instruction (launch or install) without a problem. It is assumed (to the best knowledge of the CD Launcher) that the launch or installation operation has completed successfully.
-17	The CD Launcher that is identified by `id` is no longer available. (i.e., it has become unavailable since discovery occurred).
-18	A general error has occurred. The CD Launcher knows with certainty that it has failed in its attempt to initiate the instruction (launch or install) requested.

Since there are certain actions that the CD Launcher may undertake before responding, there may be a long delay before the result is returned by the PD.

For example, the Broadcaster Application may request to launch a companion news application on a CD connected as `id` 2 based on OS or specific device type reported in the CD Launcher query API (see Section 5.8.1). The request would appear as follows:

```
--> {
    "jsonrpc": "2.0",
    "method": "org.atsc.cdApp.launch",
    "params": {
        "id": 2,
        "parameters": {
            "launch": [
               {"launchUrl": "https://www.xyza-apps.com/news.html",
                "appType": "html"}
            ]
        }
    },
    "id": 129
}
```

If the requested CD Application was successfully launched, the PD would respond with:

```
<-- {
    "jsonrpc": "2.0",
    "result": {},
    "id": 129
}
```

If the CD Launcher was not able to launch the CD Application, the PD may respond with the following results:

```
<-- {
    "jsonrpc": "2.0",
    "error": {"code": -15, "message": "User refused application launch"},
    "id": 129
}
```

Annex A: 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 application 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 application 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 application that has a text-to-voice function. Using his CD, he discovers the Spanish subtitles and uses his application 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 application. She launches that application 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 application and she can also see her ranking among other viewers who are also playing along using the tablet application.

Scenario D: George launches an On-Demand application on his main TV (primary device). The TV application requests some demographic information from George so that it can make program recommendations for George. The TV application suggests a companion tablet application that George can download to make data entry easier. George downloads and launches the tablet application. The tablet application offers George the data entry fields. George completes the data entry on his tablet and the information is registered in the TV application. The TV application 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 application 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