# ATSC Candidate Standard: Application Signaling (A/337)

Doc. S33-215r2
19 April 2017

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

This specification is being put forth as a Candidate Standard by the TG3/S33 Specialist Group. This document is a revision of the Working Draft S33-215r0 dated 20 December 2016. All ATSC members and non-members are encouraged to review and implement this specification and return comments to cs-editor@atsc.org. ATSC Members can also send comments directly to the TG3/S33 Specialist Group. This specification is expected to progress to Proposed Standard after its Candidate Standard period.

## Revision History

| Version | Date |
|---|---|
| Candidate Standard approved<br>Revision to CS approved | 19 January 2017<br>19 April 2017 |
| Standard approved | [date] |

# Table of Contents

# Index of Figures and Tables

# ATSC Candidate Standard: Application Signaling (A/337)

## 1. INTRODUCTION

### 1.1 Scope

This document specifies the signaling of application properties and synchronization of application actions with underlying audio/video content.

### 1.2 Background

This document specifies mechanisms for signaling the properties of applications, including their lifecycle states, and also mechanisms for delivering activation notifications synchronized with a time base, so that the actions of applications can be synchronized accordingly.

## 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] ATSC: "ATSC Candidate Standard: Signaling, Delivery, Synchronization and Error Protection (A/331)," Doc. S33-174r6, Advanced Television Systems Committee, Washington, D.C., 22 March 2017. *(work in progress)*

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

[3] ATSC: "ATSC Standard: Content Recovery in Redistribution Scenarios," Doc. A/336:2017, Advanced Television Systems Committee, Washington, D.C., 24 February 2017.

[4] ATSC: "ATSC Candidate Standard: Interactive Content (A/344)," Doc. S34-230r2, Advanced Television Systems Committee, Washington, D.C., 18 April 2017. *(work in progress)*

[5] ETSI: "Digital Video Broadcasting (DVB); Signaling and carriage of interactive applications and services in Hybrid broadcast/broadband environments," TS 102 809 V1.1.1, European Telecommunications Standards Institute, January 2010.

[6] ISO/IEC: "Information technology – Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats," Doc. ISO/IEC 23009-1:2014, 2nd Edition, International Organization for Standardization/International Electrotechnical Commission, Geneva Switzerland.

[7] ISO/IEC: "Information technology – Coding of audio-visual objects – Part 12: ISO base media file format," Doc. ISO/IEC 14496-12:2015, International Organization for Standardization/International Electrotechnical Commission, Geneva Switzerland.

[8] ISO/IEC: "Information technology – High efficiency coding and media delivery in heterogeneous environments – Part 1: MPEG media transport (MMT)," Doc. ISO/IEC 23008-1:201x, International Organization for Standardization/ International Electrotechnical Commission, Geneva Switzerland. *(publication expected 2017)*.

[9] IETF, RFC 6455, "The WebSocket Protocol," RFC 6455, Internet Engineering Task Force, December 2011. http://www.ietf.org/rfc/rfc6455.txt

[10] W3C: "XML Schema Part 2: Datatypes Second Edition" W3C Recommendation, Worldwide Web Consortium, 28 October 2004. https://www.w3.org/TR/xmlschema-2/

### 2.2 Informative References

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

[11] IETF: RFC 6838 (BCP 13), "Media Type Specifications and Registration Procedures," Internet Engineering Task Force, Reston, VA, January 2013. https://tools.ietf.org/html/rfc6838

## 3. TERMS

**App** – Application.

**Application** – A downloaded collection of interrelated documents intended to run in the application environment specified in the A/344, "Interactive Content" [4] and perform one or more functions, such as providing interactivity or targeted ad insertion. The documents of an application can include (but are not limited to) HTML, JavaScript, CSS, XML and multimedia files. An application can access other data that are not part of the application itself.

**Content item** – Set of one or more files which a service provider intends to be treated as a single unit for presentation purposes.

**Entry Package** – The Entry Package contains one or more files that comprise the functionality of the Broadcaster Application. The Entry Package includes the Entry Page and perhaps additional supporting files include JavaScript, CSS, image files and other content.

**Entry Page** – An initial HTML5 document referenced by application signaling that should be loaded first into the User Agent. It is included as a file within the Entry Package.

**Event** – Timed notification to receiver software or to an application indicating that some action is to be taken

**Event Stream** – Stream of events.

**Locally Cached Content Item** – A collection of one or more Locally Cached Files which are intended to be consumed as an integrated whole. A Locally Cached Content Item is typically not consumed or presented until the requisite Locally Cached Files have been fully received and cached.

**Network Content Item** – A continuous component (e.g., an audio/video clip) or a collection of one or more files (e.g., a slide show or a set of inter-linked HTML pages) that is intended to be consumed as an integrated whole, and that is delivered on demand for immediate presentation. Network Content Items are delivered via broadband and are normally progressively presented prior to receiving the entire file(s).

### 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 audio, video, and transport coding subsystems. 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.

**AEI** – Application Event Information

**DWD** – Distribution Window Description

**EA** – Emergency Alert

**ESG** – Electronic Service Guide

**HELD** – HTML Entry pages Location Description

**MPD** – Media Presentation Description

**MPU** – Media Processing Unit

**SLT** – Service List Table

**URL** – Uniform Resource Locator

**XML** – eXtensible Markup Language

## 3.4 XML Schema and Namespace

A number of new XML elements and attributes are defined and used in this Standard. These elements and attributes provide signaling of application properties defined in this standard (see for example Sections 4.2 HTML Entry pages Location Description, Section 4.3 Distribution Window Description and Section 5.1.2 Application Event Information). These new XML elements and attributes are defined with separate namespaces in schema documents that accompany this standard. The namespaces used by various schemas are described in individual sections of the present document. The sub-string part of namespaces between the right-most two ‘/’ delimiters indicate major and minor version of the schemas. The schemas defined in this present document shall have version ‘1.0’, which indicates major version is 1 and minor version is 0.

The namespace designator, “`xs:`”, and many terms in the “Data Type” column of tables is a shorthand for datatypes defined in W3C XML Schema [10] and shall be as defined there.

In order to provide flexibility for future changes in the schema, decoders of XML documents with the namespaces defined in the present document should ignore any elements or attributes they do not recognize, instead of treating them as errors.

All element groups and attribute groups are explicitly extensible with elements and attributes respectively. Elements can only be extended from namespaces other than the target namespace. Attributes can be extended from both the target namespace and other namespaces. If the XML schema does not permit this for some element, that is an error in the schema.

XML schemas shall use `processContents="strict"` in order to reduce inadvertent typos in instance documents.

In the event of any discrepancy between the XML schema definitions implied by the tables that appear in this document and those that appear in the XML schema definition files, those in the XML schema definition files are authoritative and take precedence.

The XML schema document for the schemas defined in this document can be found at the ATSC website.

## 4. SIGNALING OF APPLICATION PROPERTIES

A service can contain zero or more app-based enhancements. For example, a linear service could contain one app-based enhancement consisting of an app that runs in the background and manages the insertion of targeted ads, and another app-based enhancement that contains a collection of apps that provide an interactive viewing experience to enhance the audio/video program. Each app-based enhancement is separately signaled, so that the creators of diverse apps do not need to coordinate their signaling.

The URL of an Entry Package containing an Entry Page can be signaled for a Service, i.e., the `@entryPackageUrl` attribute in HELD defined in Section 4.2. This entry URL can change from time to time. Changes to the entry URL can be very infrequent. For example, an Entry Page contains a container (e.g., an iFrame) into which it can load different sub-pages sequentially (usually driven by Events in the broadcast). In this case the entry URL would typically change only when a new version of the Entry Page is desired. Alternatively, changes to the entry URL can be frequent. For example, an Entry Page routinely changes at show and/or interstitial boundaries. The approach to changes of the entry URL is determined by the broadcaster.

### 4.1 Application Lifecycle

The application lifecycle is indicated by the presence or absence of a valid HELD and the entries in it. This section informatively describes the application life cycle. For specific syntax and semantics of the HELD, see Section 4.2.

- Receipt of a valid HELD is an indication to a receiver that an Entry Package accessible via the `@entryPackageUrl` attribute in the HELD is available to be loaded immediately or according to the `@validFrom` and `@validUntil` attributes, if present.
- If a new `@entryUrl` attribute is signaled via HELD, it is an indication to the receiver to unload the previous Entry Page and load the new one accessible via the new `@entryUrl`

attribute immediately or according to the `@validFrom` and `@validUntil` attributes, if present.
  - o Note that logic within a given Entry Page may load and unload sub-pages as described in Section 4 above; such actions are application-specific and are not governed by the HELD.
- If the date/time indicated by the `@validUntil` attribute in HELD is reached and no new valid `@entryUrl` attribute has been signaled, it is an indication to the receiver to unload the previous Entry Page and not load a new Entry Page.
- If the HELD is no longer present or no longer valid, it is an indication to the receiver to unload the previous Entry Page and not launch a new Entry Page.

## 4.2 HTML Entry pages Location Description (HELD)

### 4.2.1 Semantics of HELD

The HELD shall be represented as an XML document containing a **`HELD`** root element that conforms to the definitions in the XML schema that has namespace:

```
tag:atsc.org,2016:XMLSchemas/ATSC3/AppSignaling/HELD/1.0/
```

The XML schema “xmlns” short name should be `"held"`.

Table 4.1 provides an informative description of the semantics of the HELD. The normative XML schema for HELD shall be as specified in the file, HELD-1.0-20170414.xsd. The normative semantics of the elements and attributes of the HELD follow the Table 4.1.

**Table 4.1** HTML Entry pages Location Description (HELD) Semantics

| Element Name | Card-inality | Data Type | Description |
|---|---|---|---|
| **HELD** | 1 | | Includes HTML entry page collection elements. |
| **HTMLEntryPackage** | 1..N | | Contains properties of the Entry Package. |
| @appContextId | 1 | anyURI | Defines the Application Context Identifier for this Entry Package. |
| @requiredCapabilities | 0..1 | sa:CapabilitiesType | Device capabilities needed for meaningful rendition of the Entry Page (as defined in the A/332, "Service Announcement" ) |
| @appRendering | 0..1 | boolean | For a linear service, indicates a broadcaster request that the broadcaster application be allowed to render the presentable component(s) of the Service |
| @entryPackageUrl | 1 | anyURI | URL of the Entry Package containing the Entry Page of the application |
| @entryUrl | 1 | anyURI | URL of the Entry Page of the application |
| @alternateEntryPkgUrl | 0..1 | anyURI | An alternate broadband path to the same Entry Package indicated in @entryPackageUrl. |
| @validFrom | 0..1 | dateTime | Indicates that the Entry Page contained in the Entry Package is to be loaded at the date and time of @validFrom, or at any time after the date and time of @validFrom and before the date and time of @validUntil when the service is selected. |
| @validUntil | 0..1 | dateTime | Indicates that the Entry Page contained in the Entry Package is to be unloaded at the date and time of @validUntil. |
| @coupledServices | 0..1 | held:listOfUnsignedShort | Provides a space-separated list of linear services sharing a common broadcaster application. |
| @lctTSIRef | 0..1 | held:listOfUnsignedInt | A list of one or more TSI values for an LCT channel which carries an application Entry Package. |

**HELD** – This root element contains one or more **HTMLEntryPage** elements.

**HTMLEntryPackage** – Each instance of this element contains information about one Entry Package that contains an HTML Entry Page.

@appContextId – This required xs:anyURI attribute represents an Application Context Identifier as a URI that indicates possible shared use of application resources among multiple Broadcaster Applications. Resources associated with a Broadcaster Application and hence an Application Context Identifier shall be made available to another Broadcaster Application if and only if each has the same Application Context Identifier. Refer to the A/344, "Interactive Content" [4] for the definition of a Broadcaster Application and how absolute URLs are formed from the relative URLs associated with broadcast-delivered resources based on the value of the @AppContextId.

Resources are delivered over the LCT channel(s) identified by the `@lctTSIRef` attribute and are made available to only the Broadcaster Applications associated with the particular Application Context Identifier.

For a given **`HTMLEntryPackage`** element present in the HELD, any files delivered over the LCT channels identified by the `@lctTSIRef` attribute for that **`HTMLEntryPackage`** element instance shall be associated with the Application Context Identifier defined for the same **`HTMLEntryPackage`** instance.

`@requiredCapabilities` – When present, this optional sa:CapabilitiesType attribute shall describe additional device capabilities needed for meaningful rendition of the Entry Page contained within this Entry Package (beyond those defined in the A/344, "Interactive Content" [4]). The syntax and semantics of the `@requiredCapabilities` attribute shall be the same as the `sa:Capabilities` element specified under the Content fragment in A/332, "Service Announcement" [2]. When the `@requiredCapabilities` attribute is not present, the capabilities required to provide a meaningful rendition of the entry page shall be as defined in the A/344, "Interactive Content" [4].

`@appRendering` – This optional xs:boolean attribute shall indicate, when present and set to `"true"`, that for the associated linear service, a broadcaster's request that the broadcaster application, via the Application Media Player as defined in the A/344, "Interactive Content" [4], be allowed to render the presentable component(s)[1] of the Service. The `@appRendering` attribute shall only be present for Services for which the value of the A/331 [1] **`SLT.Service`**`@serviceCatgory` is equal to "1" (indicating a linear A/V service with an application enhancement). At the manufacturer's discretion, receivers may choose to disregard this flag and upon initial acquisition of the Service, always begin rendering such services using the Receiver Media Player as defined in the A/344, "Interactive Content" [4]. When the `@appRendering` attribute is `"false"` or not present, the broadcaster has not requested that the broadcaster application be allowed to render the presentable component(s) of the Service.

`@entryPackageUrl` – This required xs:anyURI attribute shall reference the Entry Package containing the application Entry Page. When the package is delivered by broadcast, the package referenced by the `@entryPackageUrl` attribute shall be present in the broadcast emission. When the package is delivered by broadband, the package shall be available for download from the referenced broadband server. For broadcast-delivered content, the `@entryPackageUrl` attribute shall point to an object delivered in ROUTE file mode per A/331, "Signaling, Delivery, Synchronization and Error Protection" [1]. For broadband-delivered content, the `@entryPackageUrl` attribute shall be an absolute URL. For broadcast-delivered content, the `@entryPackageUrl` shall be a relative URL formatted as described in A/331, "Signaling, Delivery, Synchronization and Error Protection" [1].

`@entryUrl` – This required xs:anyURI attribute shall indicate the URL of the application entry point, e.g. the file in the package with a `Content-Location` value matching the value given in the `@entryUrl` attribute.

`@alternateEntryPkgUrl` – This optional xs:anyURI attribute shall be a URL that points to the same package as is referenced by the `@entryPackageUrl` attribute. If the `@alternateEntryPkgUrl` attribute is present, the delivery path for the `@entryPackageUrl` attribute shall be broadcast and the delivery path for the `@alternateEntryPkgUrl` attribute shall be broadband. The

[1] A Presentable Component is a component that is presented in a continuous media stream (e.g., audio, video or closed captioning) that is intended for presentation to the user.

@alternateEntryPkgUrl attribute shall reference a file that shall be available for download from the referenced broadband server. The @alternateEntryPkgUrl attribute shall be an absolute URL.

A change in either URL shall be an indication that a new application Entry Page is available.

@validFrom – This optional xs:dateTime attribute shall indicate that, when the service is selected, the Entry Page (the page referenced by @entryUrl) contained within the Entry Package attribute is intended to be loaded starting at the indicated date and time, or at any time after the indicated date and time and before the date and time indicated by @validUntil. When this attribute is not present, the application Entry Page is valid now and the application Entry Page is intended to be loaded at the time this HELD is received. The value of the @validFrom attribute may indicate a time in the future.

@validUntil – This optional xs:dateTime attribute shall indicate that the Entry Page (the page referenced by @entryUrl) contained within the Entry Package at the @entryUrl attribute is intended to be unloaded starting at the indicated date and time. If the @validUntil attribute is not specified, the application indicated by the @entryUrl attribute is intended to stay loaded for the indefinite future. The value of the @validUntil attribute, if present, shall indicate a time in the future relative to the delivery time of this instance of the HELD, and represent a time farther in the future than the time given in the @validFrom attribute, if the @validFrom attribute is present. When this attribute is not present, the application Entry Page is valid indefinitely.

@coupledServices – This optional held:listOfUnsignedShort attribute shall consist of a space-separated list of linear Services sharing a common broadcaster application. Each Service shall be represented as a 16-bit unsigned integer value of the **SLT.Service**@serviceId The receiver may use the information in the @coupledServices attribute as a caching hint, to preferentially retain the application content in cache upon a Service change, since a tuned-to service may employ the same application.

@lctTSIRef – This optional held:listOfUnsignedInt attribute shall consist of a space-separated list of TSI values identifying an LCT channel which carries application-related files, when they are delivered over broadcast. When this attribute is not present, there is no default value.

### 4.2.2 HELD App Distribution Timing

All application files referenced in the HELD shall be available for retrieval from the Broadcast Stream or accessible from a broadband server (as appropriate to the delivery path). For example, if the HELD is updated to indicate an application Entry Package that is valid at a time six hours from now, the files associated with that application (as indicated by the @entryPackageUrl attribute) shall be available to the receiver at the time of the HELD update, even though the launch time is six hours in the future.

### 4.2.3 HELD Examples

This section provides illustrative examples of HELD instances.

The first example is the simplest. This HELD only signals one application entry point, and gives no indication of any future expiration time.

```
<HELD>
   <HTMLEntryPackage appContextId="A.xyz.com" entryPackageUrl="app"
                     entryUrl="index.html"/>
</HELD>
```

The second example illustrates several principles:

- The ability to distribute and signal applications targeted at receivers with different capabilities;
- The ability to pre-announce the timing of a URL switch, and the application resources that will be needed at the time of the switch.

```
<HELD>
    <HTMLEntryPackage appContextId="A.xyz.com" entryPackageUrl="app" entryUrl="/p1/index.html"
        validUntil="2016-07-17T09:30:47Z"/>
    <HTMLEntryPackage appContextId="A.xyz.com" entryPackageUrl="app"
        requiredCapabilities="0700" entryUrl="/p1a/index.html" validUntil="2016-07-
        17T09:30:47Z"/>
    <HTMLEntryPackage appContextId="A.xyz.com" entryPackageUrl="app" entryUrl="/p2/index.html"
        validFrom="2016-07-17T09:30:47Z" validUntil="2016-07-17T12:00:47Z"/>
    <HTMLEntryPackage appContextId="A.xyz.com" entryPackageUrl="app"
        requiredCapabilities="0700" entryUrl="/p2a/index.html" validFrom="2016-07-
        17T09:30:47Z" validUntil="2016-07-17T12:00:47Z"/>
</HELD>
```

Note that the first **HTMLEntryPackage** element contains a URL and a `@validUntil` attribute indicating a future time after which the `p1/index.html` application is no longer valid. The second instance of the **HTMLEntryPackage** element contains a URL referencing a different application (`p1a/index.html`) and a `@requiredCapabilities` attribute indicating that the receiver must support a runtime environment capability not specified in the version of the ATSC standards the receiver was built to support (code `0700`). Receivers that recognize and support the capabilities identified with the `0700` code are expected to execute `p1a/index.html` in preference to the app89p1.html application.

## 4.3 Distribution Window Description (DWD)

### 4.3.1 Overview of DWD

The DWD fragment, comprising one or more instances of the **DistributionWindow** element, indicates that one or more application-related files (any combination of an HTML5 entry page and/or other documents of a broadcaster application such as JavaScript, CSS, XML and media files) are scheduled to be delivered by ROUTE in the future. A receiver can then tune to/join the appropriate Broadcast Stream and LCT channel over which the application-related files are broadcast during the distribution window time frame to download and store that content.

It might be desirable for any collection of application files to be broadcast during multiple distribution windows, to increase the likelihood of successful reception by a receiver having interest in those files, since the receiver may be unable to tune to the appropriate broadcast stream/LCT channel during any given distribution window instance. For example, a single-tuner receiver may be in active use and tuned to a different service during a given distribution window, but the receiver may not be in active use during a later instance of a distribution window that is delivering the same content.

The set of one or more application-related files to be delivered during an instance of **DistributionWindow** element (in the interval from the **DistributionWindow**`@startTime` attribute to the **DistributionWindow**`@endTime` attribute) is identified by the value of **DistributionWindow**`@appContentLabel` attribute. The one or more application-related files delivered during the one or more instances of **DistributionWindow** element with the same label (i.e., with the same value of `@appContentLabel` but different time windows) are identical, i.e. contain the same objects.

A DWD fragment may contain distribution window instances with multiple `@appContentLabel` attribute values, for example, label(i) for time intervals ($t_1$, $t_2$), ($t_3$, $t_4$) and ($t_5$, $t_6$), label(j) for interval ($t_7$, $t_8$) and label(k) for intervals ($t_9$, $t_{10}$) and ($t_{11}$, $t_{12}$), with $t_i < t_j$ for j>i. This enables a broadcaster to offer the application-related content in multiple different distribution windows, and also allows the receiver to avoid participating in multiple distribution windows delivering the same content.

Each distribution window instance may additionally include, under each instance of the **`AppContextID`** element, an optional `@dwFilterCode` attribute which contains one or more Filter Codes. Filter Codes are integers which are unique within a given instance of **`AppContextID.`** They are created by broadcasters to represent personalization categories as defined by individual broadcaster entities. For example, different Filter Code values may be assigned to categories such as truck owner, sustaining member, or a zip code.

Filter Codes can be associated with application-related files. In ROUTE delivery, identification of application-related files associated with Filter Codes is provided by the Extended FDT's `@fileFilterCode` attribute as described in A/331, "Signaling, Delivery, Synchronization and Error Protection" [1]. In addition, the receiver can have internally-stored Filter Code values provided by broadcaster applications (using the 'Set Filters' API as described in A/344, "Interactive Content" [4]). Filter Codes associated with files can be compared with internally-stored Filter Codes to help determine whether a given file is relevant with respect to personalization.

The `@dwFilterCode` attribute is a concatenated list of all Filter Codes for application-related files that will be available during the containing distribution window. The Filter Codes in the `@dwFilterCode` attribute can be compared with the Filter Codes stored in the receiver to help the receiver determine whether to participate in (i.e. activate content reception on the corresponding BS/LCT channel during) a given distribution window instance. If one or more Filter Codes in the `@dwFilterCode` attribute match one or more internally-stored Filter Code, then the receiver can determine that at least one file is relevant with respect to personalization.

If no Filter Codes are associated with a distribution window instance, i.e., absence of `@dwFilterCode` attribute and/or there are no internally-stored Filter Codes available in the receiver, the receiver may nonetheless participate in the distribution window and download application-related files. However, the use of Filter Codes can optimize receiver memory space by avoiding storing irrelevant material, thus allowing more space for relevant material.

### 4.3.2 Semantics of Distribution Window Description (DWD)

The DWD shall be represented as an XML document containing a **`DWD`** root element that conforms to the definitions in the XML schema that has namespace:

```
tag:atsc.org,2016:XMLSchemas/ATSC3/AppSignaling/DWD/1.0/
```

The XML schema xmlns short name should be `"dwd"`.

Table 4.2 provides an informative description of the semantics of the Distribution Window Description. The normative XML schema for DWD shall be as specified in the file, DWD-1.0-20170414.xsd. The normative semantics of the elements and attributes of the DWD follow the Table 4.2.

**Table 4.2** Distribution Window Description (DWD) Semantics

| Element Name | Card-inality | Data Type | Description |
|---|---|---|---|
| **DWD** | 1 | | |
| **DistributionWindow** | 1..N | | Broadcast transmission time frame(s) of application-related files. |
| @appContentLabel | 0..1 | unsignedInt | A label or alias for the application-related file(s) delivered during an instance of **DistributionWindow** element. Distribution Window instances identified by the same value of the @appContentLabel attribute shall transmit the same application-related file(s). |
| @startTime | 1 | dateTime | Start time of the parent instance of **DistributionWindow** element |
| @endTime | 1 | dateTime | End time of the parent instance of **DistributionWindow** element |
| @lctTSIRef | 1 | dwd:listOfUnsignedInt | A space-separated list of TSI value of the LCT channel in which the application-related files are delivered, during the parent instance of **DistributionWindow** element |
| **AppContextId** | 1..N | anyURI | Defines the Application Context Identifier for this set of Distribution Window Filter Codes. |
| @dwFilterCode | 0..1 | dwd:listOfUnsignedInt | A space-separated list of unsigned integers associated with application content item(s) to be broadcast during the affiliated instance of **DistributionWindow** element. |

**DWD** – This root element contains one or more **DistributionWindow** elements.

**DistributionWindow** – Each instance of this element shall define a single time interval during which application-related files will be sent, in the LCT channel as identified by @lctTSIRef attribute.

@appContentLabel – This optional xs:unsignedInt attribute represents a label or alias for the application-content file(s) to be delivered during an instance of **DistributionWindow** element as defined by its start and end times. Each of the one or more instances of **DistributionWindow** element identified by the same value of the @appContentLabel attribute shall transmit the same application-related file(s) between its start and end times. The uniqueness of this attribute is scoped by the broadcaster application to which this instance of **DistributionWindow** element pertains, within the interval ($t_1$, $t_2$), where $t_1$ represents the start time of the first occurrence of **DistributionWindow** with this @appContentLabel attribute value, and $t_2$ represents the expiration of this broadcaster application (i.e., the **HTMLEntryPage**@validUntil attribute of the application's entry page).

@startTime – This required xs:dateTime attribute shall specify the start time of the associated distribution window.

@endTime – This required xs:dateTime attribute shall specify the end time of the associated distribution window. The @endTime attribute should indicate a date/time in the future relative to the time this instance of the DWD is inserted into the signal. (Time-shifted content, e.g., played back from a DVR, may have an @endTime attribute that is in the past.)

`@lctTSIRef` – This required dwd:listOfUnsignedInt attribute shall specify the space-separated list of TSI values of the LCT channel in which the application-related files shall be delivered, during the parent instance of **`DistributionWindow`** element.

**`AppContextId`** – This xs:anyURI element represents an Application Context Identifier as a URI that indicates possible shared use of application resources among multiple Broadcaster Applications. Resources associated with a Broadcaster Application and hence an Application Context Identifier shall be made available to another Broadcaster Application if and only if each has the same Application Context Identifier. Refer to the A/344, “Interactive Content” [4] for the definition of a Broadcaster Application.

`@dwFilterCode` – This optional dwd:listOfUnsignedInt attribute is a space-separated list of broadcaster application specific 32-bit unsigned integers associated with the application content item(s) to be broadcast during the affiliated instance of **`DistributionWindow`** element that are scoped by the parent **`AppContextId`** element. The `@dwFilterCode` attribute may be used by the receiver platform to determine whether content available during the given Distribution Window will be of interest to the device.

### 4.3.3 DWD Examples

This section provides illustrative examples of DWD instances.

The first example is the simplest. This DWD only signals one distribution window with the start time, the end time of it, one and more LCT channel(s) and application context identifier(s).

```
<DWD>
	<DistributionWindow startTime="2016-07-17T12:00:47Z" endTime="2016-07-18T12:00:47Z" 
lctTSIRef="43 44">
		<AppContextId>A.xyz.com</AppContextId>
	</DistributionWindow>
</DWD>
```

The second example illustrates several principles:

- The ability to signal multiple distribution window with different time slot information;
- The ability to signal the label for the content objects planned to be delivered in the different time slot.

```
<DWD>
	<DistributionWindow appContentLabel="1" startTime="2016-07-17T12:00:47Z" 
endTime="2016-07-18T12:00:47Z" lctTSIRef="43 44">
		<AppContextId>A.xyz.com</AppContextId>
	</DistributionWindow>
	<DistributionWindow appContentLabel="2" startTime="2016-07-18T12:00:47Z" 
endTime="2016-07-19T12:00:47Z" lctTSIRef="43 44">
		<AppContextId>A.xyz.com</AppContextId>
	</DistributionWindow>
	<DistributionWindow appContentLabel="1" startTime="2016-07-19T12:00:47Z" 
endTime="2016-07-20T12:00:47Z" lctTSIRef="43 44">
		<AppContextId>A.xyz.com</AppContextId>
	</DistributionWindow>
		<DistributionWindow appContentLabel="2" startTime="2016-07-20T12:00:47Z" 
endTime="2016-07-21T12:00:47Z" lctTSIRef="43 44">
		<AppContextId>A.xyz.com</AppContextId>
	</DistributionWindow>
</DWD>
```

The third example illustrates several principles:

- The ability to signal the Filter Codes of app-related files distributed during available time slot for each distribution window instance.

```
<DWD>
        <DistributionWindow appContentLabel="1" startTime="2016-07-17T12:00:47Z"
endTime="2016-07-18T12:00:47Z" lctTSIRef="43 44">
                <AppContextId dwFilterCode="1 2 3" >A.xyz.com</AppContextId>
        </DistributionWindow>
        <DistributionWindow appContentLabel="1" startTime="2016-07-18T12:00:47Z"
endTime="2016-07-19T12:00:47Z" lctTSIRef="43 44">
                <AppContextId dwFilterCode="1 2 3" >A.xyz.com</AppContextId>
        </DistributionWindow>
        <DistributionWindow appContentLabel="2" startTime="2016-07-18T12:00:47Z"
endTime="2016-07-19T12:00:47Z" lctTSIRef="45 46">
                <AppContextId dwFilterCode="4 5 6" >A.xyz.com</AppContextId>
        </DistributionWindow>
</DWD>
```

### 4.4 Broadcast Delivery of HELD and DWD

When an HELD or a DWD is delivered via broadcast, it shall be delivered in the service layer signaling (SLS) for the service as defined in the A/331, “Signaling, Delivery, Synchronization and Error Protection” [1].

### 4.5 Broadband Delivery of HELD and DWD

An HELD or a DWD may be delivered via broadband using HTTP. When delivered via broadband, the HELD or DWD shall be available by an HTTP Request, using a URL for this purpose which is signaled in the SLT for the service as defined in the A/331, “Signaling, Delivery, Synchronization and Error Protection” [1] or a URL obtained from a watermark in a redistribution scenario as described in the A/336, “Content Recovery in Redistribution Scenarios” [3].

The timing and location information for retrieving a scheduled update to an HELD or DWD via broadband are provided by the `@validUntil` and `@nextURL` attributes, respectively, of the metadata envelope of the HELD as defined in the A/331, “Signaling, Delivery, Synchronization and Error Protection” [1]. An unscheduled occurrence of the availability of an updated HELD or DWD is signaled asynchronously via a dynamic event as described in Section 5.4.

## 5. SYNCHRONIZATION OF APPLICATION ACTIONS

Actions to be taken by applications can be initiated by notifications delivered via broadcast or broadband or, in a redistribution setting, via watermarks. For the purposes of this document, the term “Events” is used for such notifications.

Generically, Events are members of Event Streams. An Event Stream has the following attributes:

- schemeIdUri – globally unique identifier of the type of the Event Stream
- value – identifier of the sub-type of the Event Stream, scoped by schemeIdUri
- timescale – time scale in units per seconds to be used for deriving timing information of Events in the Event Stream

Each individual Event in an Event Stream has the following additional attributes and an element:

- presentationTime – start time of the Event relative to the start of the Period that the Event Stream is present
- duration – duration of the Event
- id – unique identifier of the Event within the Event Stream
- data (optional) – data that accompanies the Event, to be used by an application that responds to the Event

The data types of these attributes and their precise semantics differ slightly in different situations, but they all basically represent the same properties in all cases.

The specifier of an Event Stream selects the `@schemeIdUri` attribute and determines the possible values of the `@value` attribute and their properties, including whether a **`data`** element is included, and if so what its structure is.

## 5.1 Broadcast Delivery of Events

There are slight differences between the broadcast delivery of Events for ROUTE/DASH-based services and MMT-based services, but there is also significant commonality.

### 5.1.1 Broadcast Events for ROUTE/DASH-based Services

When delivered via broadcast in a ROUTE/DASH-based system, Events may be delivered as DASH Events, using either of the two mechanisms for Event delivery defined in the DASH specification:

- **`EventStream`** element(s) appearing in a **`Period`** element of the MPD
- Event(s) in `'emsg'` box(es) appearing in the Segments of the Representation, with their presence signaled by one or more **`InbandEventStream`** elements of the **`Representation`** element in the MPD

These two delivery mechanisms may be mixed. A single Event Stream may include some Events delivered via an **`EventStream`** element and others delivered via `'emsg'` boxes.

The **`EventStream`** element, defined in Section 5.10.2 of the DASH standard [6], is especially well suited for "static" Events – i.e., Events for which the timing is known well ahead of time. An **`EventStream`** element is basically a list of **`Event`** elements. Each **`EventStream`** element has a `@schemeIdUri` attribute and a `@value` attribute to identify the type of Events in the **`EventStream`** element, and a `@timescale` attribute to indicate the reference time scale for the Event presentation times and durations. Each **`Event`** element in an **`EventStream`** element has a `@presentationTime` attribute to indicate the start time of the Event (relative to the start of the **`Period`** element), a `@duration` attribute to indicate the duration of the Event, an `@id` attribute to identify the Event instance, and an optional **`data`** element to provide information for carrying out the action initiated by the Event. The structure of the **`data`** element is determined by the type of the Event. There can be multiple **`EventStream`** elements of different types in a **`Period`** element.

Certain DASH-specific `@schemeIdUri` attributes are defined in Section 5.10.4 of the DASH standard [6], along with the usage of the accompanying `@value` attribute and the semantics of the corresponding events. Additional `@schemeIdUri` attributes can be defined as needed. The "owner" of a `@schemeIdUri` attribute value must ensure that it is unique (for example, that it is based on a URI controlled by the owner), and must define the usage of the corresponding `@value` attribute and the semantics of the events.

An ATSC-specific `@schemeIdUri` attribute is defined in Section 5.4 of this document, along with the usage of the accompanying `@value` attribute and the semantics of the Events.

Other @schemeIdUri attributes can be defined by application developers to meet the needs of their applications.

An **InbandEventStream** element of a Representation indicates the presence of 'emsg' boxes in the Segments of the Representation. The **InbandEvent** element has a @schemeIdUri attribute and a @value attribute to indicate the type of the Events that can appear. Each 'emsg' box that appears in a Segment of a Representation has fields schemeIdUri and value to indicate the Event Stream, they belong to, and fields (a) timescale to indicate the reference time scale for the Event, used for the presentation time and the duration, (b) presentation_time_delta to indicate the start time of the Event relative to the earliest presentation time of any access unit in the Segment in which the 'emsg' box appears, (c) event_duration to indicate the duration of the Event, (d) id to identify the Event instance, and optionally (e) message_data if needed to carry out the action initiated by the Event. Events delivered in 'emsg' boxes are especially well suited for "dynamic" Events – i.e., Events for which the timing only becomes known at the last minute (such as an Event initiating some action by an application when a team scores in a live sports program).

#### 5.1.2 Broadcast Events for MMT-based Services

When delivered via broadcast in an MMT-based system, Events may be delivered in an XML document called an Application Event Information (AEI) document. This document is especially well suited for static Events.

The AEI shall be represented as an XML document containing a **AEI** root element that conforms to the definitions in the XML schema that has namespace:

```
tag:atsc.org,2016:XMLSchemas/ATSC3/AppSignaling/AEI/1.0/
```

The XML schema xmlns short name should be "aei".

Table 5.1 below indicates the structure of the AEI. The normative XML schema for AEI shall be as specified in the file, AEI-1.0-20170414.xsd.

**Table 5.1** Syntax of the AEI

| Element Name | | | | Card-inality | Data Type | Description |
|---|---|---|---|---|---|---|
| **AEI** | | | | | | Set of static event streams |
| | @assetId | | | 1 | string | Identifier of the MMT asset used for time reference |
| | @mpuSeqNum | | | 1 | unsignedInt | Identifier of the anchor MPU used for time reference |
| | @timeStamp | | | 1 | unsignedLong | The presentation time of the anchor MPU |
| | **EventStream** | | | 1..N | | Event Stream |
| | | @schemeIdUri | | 1 | anyURI | Identifier scheme of the event stream. The string may use URN or URL syntax. |
| | | @value | | 0..1 | string | Identifier "`value`" attribute of the event stream |
| | | @timescale | | 0..1 | unsignedInt | Time scale to be used for events in the event stream |
| | | **Event** | | 1..N | string | Event envelope and string data for the event. The content of the string depends on the event scheme and value attributes. |
| | | | @presentationTime | 0..1 | unsignedLong | Presentation time of the event relative to the presentation time of the first access unit in the anchor MPU, which is indicated by `@timestamp` |
| | | | @duration | 0..1 | unsignedLong | Duration of this event. |
| | | | @id | 0..1 | unsignedInt | Identifier for this event. |

The normative semantics of elements and attributes in AEI table shall be as follows:

**`AEI`** – This root element describes a set of static event streams and contains one or more **`EventStream`** elements.

`@assetId` – This required attribute specifies identifier of the MMT asset whose MPU is used as the anchor for time reference for events in the **`EventStream`** elements. The value of this shall be equal to asset_id() value in ISO/ IEC 23008-1 [8].

`@mpuSeqNum` – This required attribute specifies sequence number of the anchor MPU in the MMT asset identified by `AEI@assetId` used as anchor for time reference for events in the **`EventStream`** elements.

`@timestamp` – This required attribute specifies the presentation time of the first access unit in the anchor MPU indicated by `AEI@mpuSeqNum` within the asset indicated by `AEI@assetId`. The format of ISO/ IEC 23008-1 [8] MPU_Timestamp_descriptor()'s mpu_presentation_time field shall be used for this attribute.

**`EventStream`** – This element and its attributes shall describe information about an event stream.

`@schemeIdUri` – This required attribute specifies an identifier scheme for this event stream. This string may use a URN or a URL syntax. Each **`AEI.EventStream`** element shall have a unique value for this attribute.

`@value` – This optional attribute specifies the value of the event stream within the scope of **`AEI.EventStream@schemeIdUri`**. When not present no default value is defined.

`@timescale` – This optional attribute specifies time scale in units per second to be used for events in this event stream. When not present **`AEI.EventStream@timescale`** is inferred to be equal to 1. **`AEI.EventStream@timescale`** shall not be equal to 0.

**`Event`** – Each instance of this element and its attributes shall define information about an event in the context of the parent event stream element.

`@presentationTime` – This optional attribute specifies presentation time of the event relative to the presentation time of the first access unit in the anchor MPU indicated with sequence number specified by the parent AEI@mpuSeqNum within the asset indicated by asset ID specified by AEI@assetId. The relative value of the presentation time in seconds is equal to AEI.EventStream.Event@presentationTime divided by AEI.EventStream@timeScale. When not present **AEI.EventStream.Event@presentationTime** is inferred to be equal to 0.

@duration – This optional attribute specifies presentation duration of the event. The presentation duration in seconds is equal to AEI.EventStream.Event@duration divided by AEI.EventStream@timeScale. When this attribute is not present no default value is inferred.

@id – This optional attribute specifies an identifier of this event within the scope of parent **AEI.EventStream@schemeIdUri** and **AEI.EventStream@value**. When this attribute is not present no default value is inferred.

When an AEI is delivered via broadcast, it shall be delivered as the payload of an mmt_atsc3_message() as defined in the section of ATSC A/331, “Signaling, Delivery, Synchronization and Error Protection” [1], that specifies the MMTP-Specific Signaling Message.

Events in an MMT-based service may also be carried in ‘evti’ boxes in MPUs. This method is especially well suited for dynamic Events. Table 5.2 below indicates the structure of an ‘evti’ box, using the usual form of specification for a box in an ISO BMFF file [7].

**Table 5.2** Syntax of an ‘evti’ Box

```
aligned(8) class EventInformationBox extends
                                FullBox(‘evti’, version = 0, flags = 0){
      string scheme_id_uri;
      string value;
      unsigned int(32) timescale;
      unsigned int(32) event_id;
      unsigned int(32) event_presentation_time_delta;
            /*relative to the earliest presentation time in this MPU */
      unsigned int(32) event_duration;
      unsigned int(8) event_data[]; }
  }
```

Such an ‘evti’ box may appear at the beginning of the MPU, after the ‘ftyp’ box, but before the ‘moov’ box, or it may appear immediately before any ‘moof’ box.

An inband_event_descriptor() indicates the presence of Events in the MPUs. The syntax of this descriptor shall be as provided in Table 5.3. The semantics of the fields in inband_event_descriptor() shall be as given immediately below the table.

**Table 5.3** Syntax of the inband_event_descriptor()

| Syntax | Value | No. of Bits | Format |
|---|---|---|---|
| inband_event_descriptor() { | | | |
| **descriptor_tag** | | 16 | uimsbf |
| **descriptor_length** | | 16 | uimsbf |
| **number_of_assets** | N1 | 8 | uimsbf |
| for (i=0;i<N1;i++) { | | | |
| **asset_id_length** | N2 | 32 | uimsbf |
| for (j=0;j<N2;j++) { | | | |
| **asset_id_byte** | | 8 | uimsbf |
| } | | | |
| **scheme_id_uri_length** | N3 | 8 | uimsbf |
| for (j=0;j<N3;j++) { | | | |
| **scheme_id_uri_byte** | | 8 | uimsbf |
| } | | | |
| **event_value_length** | N4 | 8 | uimsbf |
| for (j=0;j<N4;j++) { | | | |
| **event_value_bytes** | | 8 | uimsbf |
| } | | | |
| } | | | |
| } | | | |

**descriptor_tag** – This 16-bit unsigned integer shall have the value 0x0007, identifying this descriptor as the inband_event_descriptor().

**descriptor_length** – This 16-bit unsigned integer shall specify the length (in bytes) immediately following this field up to the end of this descriptor.

**number_of_assets** – An 8-bit unsigned integer field that shall specify the number of assets described by this descriptor.

**asset_id_length** – This 32-bit unsigned integer field shall specify the length in bytes of the asset_id.

**asset_id_byte** – An 8-bit unsigned integer field that shall contain a byte of the asset id.

**scheme_id_uri_length** – This 8-bit unsigned integer field shall specify the length in bytes of `scheme_id_uri` which identifies the scheme of the event stream.

**scheme_id_uri_byte** – An 8-bit unsigned integer field that shall contain a byte of `scheme_id_uri`.

**event_value_length** – This 8-bit unsigned integer field shall specify the length in bytes of the "value" attribute of the event stream.

**event_value_byte** – An 8-bit unsigned integer field that shall contain a byte of the "value" attribute of the event stream.

When an inband_event_descriptor() is delivered via broadcast, it shall be delivered as the payload of an mmt_atsc3_message() as defined in the section of ATSC A/331, "Signaling, Delivery, Synchronization and Error Protection" [1], that specifies the MMTP-Specific Signaling Message.

## 5.2 Broadband Delivery of Events

Just as broadcast delivery supports batch delivery of Events in an MPD or AEI and incremental delivery of Events in Representation Segments or MPUs, broadband delivery also supports batch delivery and incremental delivery.

When Events for a service are delivered via broadband in batch mode (which is especially suitable for static Events), they may be delivered in `EventStream` elements in an MPD which is delivered via broadband using HTTP, for a ROUTE/DASH streaming service, or in an AEI which is delivered via broadband using HTTP, for an MMTP/MPU streaming service. Such an AEI shall have the same structure as defined in Section 5.1.2 above. When delivered via broadband, MPDs and AEIs shall be available by an HTTP Request, using the base URL for this purpose which is signaled in the SLT for the service (or a URL obtained from a watermark in a redistribution scenario as described in [3]).

The timing and location information for retrieving a scheduled update to an MPD or AEI via broadband are provided by the `validUntil` and `nextURL` properties, respectively, in the metadata envelope of the MPD or AEI. An unscheduled occurrence of the availability of an updated MPD or AEI is signaled asynchronously via a dynamic event described in Section 5.4.

When Events for a service are delivered incrementally via broadband (which is especially suitable for dynamic Events), they may be delivered as ‘`emsg`’ boxes in DASH Segments for ROUTE-based services or as `'evti'` boxes for MMTP-based services. The ‘`emsg`’ and ‘`evti`’ boxes may be acquired via polling an HTTP server using the URL of a Signaling Server obtained from the SLT [1], or they may be acquired via a WebSocket communications using the URL of a Dynamic Event WebSocket Server obtained from the SLT [1]. The full protocol for acquiring dynamic events from a WebSocket server is specified in Section 5.5.

The format of Events delivered via HTTP or WebSocket servers shall be exactly the same as the format of the ‘`emsg`’ boxes described above for ROUTE/DASH-based services, or the format of the ‘evti’ boxes defined above for MMTP/MPU-delivered services, except that in the ROUTE/DASH case they are prefixed with an MPD ID and a Period ID, and in the MMTP/MPU case they are prefixed with an Asset ID and an MPU sequence number. In the ROUTE/DASH case, the `presentation_delta` is relative to the start time of the referenced `Period`, and in the MMTP/MPU case, the `presentation_delta` is relative to the earliest access unit presentation time of the referenced MPU.

## 5.3 Watermark Delivery of Events

In a redistribution setting, Events can be also acquired via watermarks. Events can be delivered in a video watermark, or by an Event server after a flag in an audio watermark indicates that an Event is available. These processes are described in the A/336 “Content Recovery in Redistribution Scenarios” standard [3].

## 5.4 ATSC-Specific Event Streams

There are three types of Event Streams of interest in this standard which can be delivered via any mechanism described in Section 5:

- DASH-specific signaling Event Streams, defined in Section 5.10.4 of the DASH standard [6]
- Application-specific Event Streams, defined by application developers
- ATSC-specific signaling Event Streams, defined below

The DASH-specific signaling Event Streams may be used as specified in the DASH standard [6].

Application-specific Event Streams are defined by application developers. The only constraints are that the combination of `@schemeIdUri/@value` attributes must be globally unique, such as by the use of a `@schemeIdUri` attribute controlled by the application developer, and by proper management of the `@value` attribute. In order to get access to these Events, applications register callback routines for them, and the callback routines are called when such Events arrive.

ATSC-specific signaling Events are used to notify devices when unexpected updates to signaling metadata objects become available. The `@schemeIdUri` attribute for an ATSC-specific Event Stream shall be of form "`tag:atsc.org,2016:event`", and the `@value` attribute for Event Streams that announce ATSC 3.0 signaling metadata object updates shall be "`stu`". The **`data`** element shall be a comma separated list of the updated table name(s), where the allowed table names shall be the individual signaling metadata object names listed in the table for the supported types of metadata objects in the section of A/331, "Signaling, Delivery, Synchronization and Error Protection" [1], that describes how signaling metadata objects can be used to make HTTP requests to the signaling server.

## 5.5 Event notification via WebSocket

### 5.5.1 Introduction

Various dynamic events could be delivered by broadband in addition to broadcast. Since new event information may need to be communicated dynamically at any time, use of notification is supported for broadband delivery of dynamic events.

The following types of dynamic notification of events are supported over broadband.

- Notification about availability of an instance of event information for a service
- Notification about availability of an instance of event information for a service along with the inclusion of signaling object data in the notification

### 5.5.2 Dynamic Notification

The following describes the steps taken for dynamic notification of event information over a broadband connection.

1) Broadband server URL from where dynamic event notifications can be received is signaled in the Broadcast Stream in the Service List Table (SLT) [1].
2) A WebSocket connection is established by the client with an event notification URL server as per RFC 6455 [9] for receiving event notification (and optionally signaling object data notification) messages. Signaling object data includes such data as present in the HELD, MPD, or AEI.

A WebSocket subprotocol `EventNotify` as defined below shall be used for this.

The opening handshake for this between the client and the server is as shown below.

The HTTP upgrade request from client to server is as follows:

```
GET /notifications HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: ePhhsdhjdshuwrwrrwjQDS==
Origin: http://server.com
Sec-WebSocket-Protocol: EventNotify
Sec-WebSocket-Version: 13
NotificationType:
```

The successful response from server to client is as follows:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: 6d67dfukfhwHGJHOwqQEE+kjfh=
Sec-WebSocket-Protocol: EventNotify
NotificationType: 1
```

#### 5.5.2.1 NotificationType Extension for Sec-WebSocket-Extension Header Field

A `Sec-WebSocket-Extensions` header field extension termed `NotificationType` is defined. An `extension-para` is defined for the `NotificationType` extension with valid values of 0 and 1, i.e. `ntval=(0|1)`. `NotificationType` extension can be used in `Sec-WebSocket-Extension` request header and `Sec-WebSocket-Extension` response header. When used in `Sec-WebSocket-Extension` request header `NotificationType` extension indicates if only event information availability notification is requested (value of 0 for `ntval extension-param`) or event information availability notification along with signaling object data is requested (value of 1 for `ntval extension-param`). When used in `Sec-WebSocket-Extensions` response header `NotificationType` extension indicates if only event information availability notification is sent in the notification response (value of 0 for `ntval extension-param`) or event information availability notification along with signaling object data is sent in the notification response (value of 1 for `ntval extension-param`).

- If the server supports sending an event information availability notification along with signaling object data in the notification message and if the request from the client includes

  ```
  Sec-WebSocket-Extensions: NotificationType; ntval=1
  ```

  header then the server shall respond with a

  ```
  Sec-WebSocket-Extensions: NotificationType; ntval=1
  ```

  header in the response and shall send notification messages using the EventNotify subprotocol described below with non-zero OBJECT_DATA length.
- If the server supports sending an event information availability notification along with signaling object data in the notification message and if the request from the client includes

  ```
  Sec-WebSocket-Extensions: NotificationType; ntval=0
  ```

  header then the server shall respond with

  ```
  Sec-WebSocket-Extensions: NotificationType; ntval=0
  ```

  header in the response and shall send notification messages using EventNotify subprotocol described below with zero OBJECT_DATA length and not including signaling object data in the notification message.
- If the server does not support sending signaling object data along with the event information availability notification in the notification message and if the request from the client includes

  ```
  Sec-WebSocket-Extensions: NotificationType; ntval=1
  ```

  header then the server shall respond with

```
Sec-WebSocket-Extensions: NotificationType; ntval=0
```

header in the response and shall send notification messages using EventNotify subprotocol described below with zero OBJECT_DATA length and not including signaling object data in the notification message.

- If the server does not support sending signaling object data along with the event information availability notification in the notification message and if the request from the client includes

  ```
  Sec-WebSocket-Extensions: NotificationType; ntval=0
  ```

  header then the server shall respond with

  ```
  Sec-WebSocket-Extensions: NotificationType; ntval=0
  ```

  header in the response and shall send notification messages using EventNotify subprotocol described below with zero OBJECT_DATA length and not including signaling object data in the notification message.

#### 5.5.2.2 EventNotify Subprotocol

The `EventNotify` subprotocol framing structure is shown in Figure 5.1. Table 5.3 describes the elements in the `EventNotify` framing structure along with their semantics. `EventNotify` protocol shall use the WebSocket ‘binary’ format with Opcode %x2 for base framing (or %x0 for continuation frame) for the messages.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         NOTIFY_ID             |         SERVICE_ID            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   AC  |  ET   |  EF |EE |      DATA_LENGTH                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    EVENT_INFORMATION ...                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       OBJECT_LENGTH           |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                        OBJECT DATA                            |
|                           ...                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**Figure 5.1** `EventNotify` subprotocol framing structure.

**Table 5.3** `EventNotify` Subprotocol Framing Elements

| Element | No. of Bits | Semantics |
|---|---|---|
| NOTIFY_ID | 16 | A notification identifier which uniquely identifies this event notification connection.<br>NOTIFY_ID values in the range of 0xF000-0xFFFF are reserved for action code value of 2 and 3. |
| SERVICE_ID | 16 | Service identifier for which the notification and the table data is applicable. SERVICE_ID uniquely identifies the service. This shall correspond to serviceId attribute value for a service in service list table (SLT) as specified in [1]. |
| AC (ACTION_CODE) | 4 | Defines the type of action. Following actions are defined.<br>0: Event notification from server to client.<br>1: Event notification PAUSE request from client to server events for the service identified by SERVICE_ID for the notification connection identified by NOTIFY_ID.<br>2: Event notification RESUME request from client to server events for the service identified by SERVICE_ID for the notification connection identified by NOTIFY_ID.<br>3: Request from client to server to receive current event information for the service identified by SERVICE_ID field. The field NOTIFY_ID provides identifier for this request.<br>4: Event information response from server to client for the request from the client. The server shall use the same NOTIFY_ID field as the request from the client.<br>5-15 : reserved. |
| ET | 4 | Indicates the type of event for which updated data is available. ET field value is interpreted as follows:<br>0 indicates ATSC ROUTE/DASH event.<br>1 indicates ATSC MMT event.<br>2-15 are reserved. |
| EF | 3 | Defines format of OBJECT_DATA. Following formats are defined.<br>0: Binary format used for OBJECT_DATA.<br>1: XML format used for OBJECT_DATA.<br>2: JSON format used for OBJECT_DATA.<br>3-7: reserved. |
| EE | 2 | Defines encoding for OBJECT_DATA. Following encodings are defined.<br>0: No encoding.<br>1: GZIP encoding as per RFC 1952.<br>2-3: reserved. |
| DATA_LENGTH | 17 | Provides the length in bytes of EVENT_INFORMATION data that follows. If the DATA_LEGNTH is zero then the EVENT_INFORMATION is not included in the event notification.<br>DATA_LENGTH shall be zero when AC field has a value in the range of 1 to 3, inclusive. |
| EVENT_INFORMATION | DATA_LENGH | Event information for the event.<br>When ET (EVENT_TYPE) is 0 the EVENT_INFORMATION content will be same as content of ‘emsg’ box as specified in ISO/ IEC 23009-1 [6].<br>When ET (EVENT_TYPE) is 1 the EVENT_INFORMATION content will be same as content of ‘evti’ box. |
| OBJECT_LENGTH | 16 | Provides the length in bytes of OBJECT_DATA field. If the OBJECT_LENGTH is zero then the OBJECT_DATA is not included in the notification.<br>OBJECT_LENGTH shall be zero when AC field has a value in the range of 1 to 3, inclusive. |

| | | |
|---|---|---|
| OBJECT_DATA | OBJECT_LENGH | Optional signaling object data corresponding to this event. The OBJECT_DATA shall have the syntax based on value of other fields in this event.<br>The signaling object data can be in bitstream (binary) or XML or JSON format. Rules specified in the NotificationType header field regarding inclusion of OBJECT_DATA element shall be followed. |

(1) When a new dynamic event needs to be notified, the server shall notify it to the client within 10 seconds over the established WebSocket connection using `EventNotify` subprotocol with AC (`ACTION_CODE`) value of 0.

(2) Pausing/ resuming receiving ATSC event notifications for a service:

The client receiving notifications can pause receiving notifications for particular service identified by `SERVICE_ID` by sending AC (`ACTION_CODE`) value of 1 in the `EventNotify` message to the server.

Upon receiving such a `PAUSE` message the server shall pause sending events to the client on the notification stream identified by the `NOTIFY_ID` field in the client request for the type of event identified by the `ET` field in the client request for the service identified by the `SERVICE_ID` field in the client request.

The client that was previously receiving events which it has paused can resume receiving notifications for a particular event type identified by `ET` for particular service identified by `SERVICE_ID` by sending `AC` (`ACTION_CODE`) value of 2 in the `EventNotify` message to the server.

Upon receiving such a `RESUME` message the server shall resume sending events to the client on the notification stream identified by the `NOTIFY_ID` field in the client request for the type of event identified by the `ET` field in the client request for the service identified by the `SERVICE_ID` field in the client request if the events were previously paused.

(3) Request/ Response support for application/ event table retrieval for a service:

The client can send a request to receive the current table by sending AC (`ACTION_CODE`) value of 3 for a particular event type identified by `ET` for a particular service identified by `SERVICE_ID` in the `EventNotify` message to the server. In this case the client will randomly assign a `NOTIFY_ID` value in the range of 0xF000 to 0xFFFF to identify the request.

Upon receiving such a request message the server shall send the current event to the client for the type of event identified by the `ET` field in the client request for the service identified by the `SERVICE_ID` field in the client request with `NOTIFY_ID` field set to the value included in the client request.

(4) The WebSocket connection can be closed from either server or client at any time.

## 6. DELIVERY OF LOCALLY CACHED CONTENT ITEMS

Delivery of Locally Cached content items consumed by an application shall conform to the A/331, “Signaling, Delivery, Synchronization and Error Protection” [1], except that an HELD or a DWD may be used to identify content items to be delivered. Moreover, applications can initiate the broadband delivery of Locally Cached content items or files even when they are not listed in an HELD or DWD.

## 7. DELIVERY OF NETWORK CONTENT

Network content items shall be delivered via broadband, using HTTP/HTTPS, with the delivery to be initiated by applications.

# ***Annex A***: Media Type Registrations

This Annex registers new media types. *Notice to editors: any changes to this Annex are subject to review by IETF and IANA as described in IETF BCP 13[11].*

## A.1 HELD

Type name:

application

Subtype name:

atsc-held+xml

Required parameters:

None.

Optional parameters:

charset

If specified, the charset parameter must match the XML encoding declaration, or if absent, the encoding is determined from the XML document itself. See also “Encoding considerations” below.

Encoding considerations:

Same for application/xml, constrained to UTF-8. See IETF 7303, Section 9.1. For the purpose of filling out the IANA Application for Media Type, the value, “binary”, applies.

Security considerations:

This media format is used to describe broadcast and broadband services. This format is highly susceptible to manipulation or spoofing for attacks desiring to mislead a receiver about a session. Both integrity protection and source authentication is recommended to prevent misleading of processors.

Interoperability considerations:

The published specification describes processing semantics that dictate behavior that must be followed when dealing with, among other things, unrecognized elements and attributes, both in the document’s namespace and in other namespaces.

Because this is extensible, conformant processors may expect (and enforce) that content received is well-formed XML, but it cannot be guaranteed that the content is valid to a particular DTD or Schema or that the processor will recognize all of the elements and attributes in the document.

Published specification:

This media type registration is an integral part of ATSC A/337, “Application Signaling”, Annex A. The payload is defined in Section 4.2.1.

Applications that use this media type:

ATSC 3.0 television and Internet encoders, decoders and other facility and consumer equipment.

Additional information:

File extension(s):

.held

Macintosh file type code(s):

"HELD"

Person & email address to contact for further information:

Editor, Advanced Television Systems Committee (jwhitaker@atsc.org)

Intended usage:

COMMON

Restrictions on usage:

None

Author:

ATSC.

Change controller:

ATSC.

## A.2 DWD

Type name:

application

Subtype name:

atsc-dwd+xml

Required parameters:

None.

Optional parameters:

charset

If specified, the charset parameter must match the XML encoding declaration, or if absent, the encoding is determined from the XML document itself. See also “Encoding considerations” below.

Encoding considerations:

Same for application/xml, constrained to UTF-8. See IETF 7303, Section 9.1. For the purpose of filling out the IANA Application for Media Type, the value, “binary”, applies.

Security considerations:

This media format is used to describe broadcast and broadband services. This format is highly susceptible to manipulation or spoofing for attacks desiring to mislead a receiver about a session. Both integrity protection and source authentication is recommended to prevent misleading of processors.

Interoperability considerations:

The published specification describes processing semantics that dictate behavior that must be followed when dealing with, among other things, unrecognized elements and attributes, both in the document’s namespace and in other namespaces.

Because this is extensible, conformant processors may expect (and enforce) that content received is well-formed XML, but it cannot be guaranteed that the content is valid to a particular DTD or Schema or that the processor will recognize all of the elements and attributes in the document.

Published specification:

This media type registration is an integral part of ATSC A/337, "Application Signaling", Annex A. The payload is defined in Section 4.3.2.

Applications that use this media type:

ATSC 3.0 television and Internet encoders, decoders and other facility and consumer equipment.

Additional information:

File extension(s):

.dwd

Macintosh file type code(s):

"DWD"

Person & email address to contact for further information:

Editor, Advanced Television Systems Committee (jwhitaker@atsc.org)

Intended usage:

COMMON

Restrictions on usage:

None

Author:

ATSC.

Change controller:

ATSC.

### A.3 AEI

Type name:

application

Subtype name:

mmt-aei+xml

Required parameters:

None.

Optional parameters:

charset

If specified, the charset parameter must match the XML encoding declaration, or if absent, the encoding is determined from the XML document itself. See also "Encoding considerations" below.

Encoding considerations:

Same for application/xml, constrained to UTF-8. See IETF 7303, Section 9.1. For the purpose of filling out the IANA Application for Media Type, the value, "binary", applies.

Security considerations:

This media format is used to describe broadcast and broadband services. This format is highly susceptible to manipulation or spoofing for attacks desiring to mislead a receiver about a session. Both integrity protection and source authentication is recommended to prevent misleading of processors.

Interoperability considerations:

The published specification describes processing semantics that dictate behavior that must be followed when dealing with, among other things, unrecognized elements and attributes, both in the document's namespace and in other namespaces.

Because this is extensible, conformant processors may expect (and enforce) that content received is well-formed XML, but it cannot be guaranteed that the content is valid to a particular DTD or Schema or that the processor will recognize all of the elements and attributes in the document.

Published specification:

This media type registration is an integral part of ATSC A/337, "Application Signaling", Annex A. The payload is defined in Section 5.1.2.

Applications that use this media type:

ATSC 3.0 television and Internet encoders, decoders and other facility and consumer equipment.

Additional information:

File extension(s):

.maei

Macintosh file type code(s):

"MAEI"

Person & email address to contact for further information:

Editor, Advanced Television Systems Committee (jwhitaker@atsc.org)

Intended usage:

COMMON

Restrictions on usage:

None

Author:

ATSC.

Change controller:

ATSC.

End of Document