ATSC Standard:
Application Event Delivery

Doc. A/337:2024-04
3 April 2024

Advanced Television Systems Committee
1300 I Street, N.W., Suite 400E
Washington, D.C. 20005
202-872-9160

The Advanced Television Systems Committee, Inc. is an international, non-profit organization developing voluntary standards and recommended practices for broadcast television and multimedia data distribution. ATSC member organizations represent the broadcast, professional equipment, motion picture, consumer electronics, computer, cable, satellite, and semiconductor industries. ATSC also develops 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 Consumer Technology Association (CTA), the Institute of Electrical and Electronics Engineers (IEEE), the National Association of Broadcasters (NAB), the Internet & Television 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	19 January 2017
Update 1 to CS approved	19 April 2017
Update 2 to CS approved	12 July 2017
Update 3 to CS approved	31 October 2017
A/337:2018 Standard approved	2 January 2018
Candidate Standard Revision of A/337:2018 approved	21 November 2018
Update to CS approved	20 February 2019
A/337:2019 Standard approved	30 April 2019
Hyperlink in referend [4] corrected	5 March 2020
A/337:2022-03 (references to ATSC documents updated)	31 March 2022
A/337:2023-03 (references to ATSC documents updated)	28 March 2023
A/337:2024-04 (references to ATSC documents updated)	3 April 2024

Table of Contents

Index of Figures and Tables

ATSC Standard:
Application Event Delivery

1. INTRODUCTION

1.1 Scope

This document specifies the delivery of events/actions for ATSC 3.0 applications and synchronization of these application events/actions with underlying audio/video content. See also A/331 [1] for details about application signaling and A/344 [11] for details about the application runtime environment.

1.2 Background

This document specifies 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 Standard: Signaling, Delivery, Synchronization and Error Protection," Doc. A/331:2024-04, Advanced Television Systems Committee, Washington, DC, 3 April 2024.

[2] ATSC: "ATSC Standard: Service Announcement," Doc. A/332:2024-04, Advanced Television Systems Committee, Washington, DC, 3 April 2024.

[3] ATSC: "ATSC Standard: Content Recovery in Redistribution Scenarios," Doc. A/336:2024-04, Advanced Television Systems Committee, Washington, DC, 3 April 2024.

[4] DASH IF: "Guidelines for Implementation: DASH-IF Interoperability Points for ATSC 3.0, Version 1.0," DASH Interoperability Forum, January 31, 2017. https://dashif.org/guidelines/

[5] 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.

[6] 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.

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

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

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

[10] IETF, RFC 7231, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content,", Internet Engineering Task Force, June 2014.
http://www.ietf.org/rfc/rfc7231.txt

2.2 Informative References

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

[11] ATSC: "ATSC Standard: Interactive Content," Doc. A/344:2024-04," Advanced Television Systems Committee, Washington, DC, 3 April 2024.

[12] 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

[13] IETF: RFC 7303, "XML Media Types," Internet Engineering Task Force, Reston, VA, July 2014.
https://tools.ietf.org/html/rfc7303

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" [11] 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.

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

Event Stream – Stream of events.

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 Abbreviations

The following acronyms and abbreviations are used within this document.

AEI – Application Event Information
EA – Emergency Alert
ESG – Electronic Service Guide
HELD – HTML Entry pages Location Description
LCT – Layered Coding Transport
MPD – Media Presentation Description
MPU – Media Processing Unit
SLT – Service List Table
TSI – Transport Session Identifier
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 Section 4.1.2 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 [9] 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.

XML instance documents shall use UTF-8 encoding.

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. 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.

4.1 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.

4.1.1 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, of which background and basic use cases are described in Section 5.6.3 of the DASH-IF IOP for ATSC 3.0 [4],:

- **`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 **`AdaptationSet`** or **`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 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.

The constraints and extensions of the existing DASH-specific Events are defined in Section 5.6.2 and 5.6.3 of the DASH-IF IOP for ATSC 3.0 [4].

An ATSC-specific @schemeIdUri attribute is defined in Section 4.4 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).

4.1.2 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 4.1 below indicates the structure of the AEI. The normative XML schema for AEI shall be as specified in the file, AEI-1.0-20171016.xsd.

Table 4.1 Syntax of the AEI

Element Name	Use	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 [7].

@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 [7] 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. This element includes the data corresponding to the event coded as a XML string.

@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 4.2 below indicates the structure of an 'evti' box, using the usual form of specification for a box in an ISO BMFF file [6].

Definition of 'evti' box:

- Box Type: 'evti'
- Container: MPU
- Mandatory: No
- Quantity: Zero or more

Table 4.2 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;
      unsigned int(32) event_duration;
      unsigned int(8) event_data[]; }
   }
```

The normative semantics of elements in 'evti' box shall be as follows:

scheme_id_uri – This field specifies an identifier scheme for this event. This string may use a URN or a URL syntax. Multiple'evti' boxes with same scheme_id_uri may be present.

value – This field specifies the value for this event within the scope of scheme_id_uri.

timescale – This field provides the time scale, in units per second to be used for this event. timescale shall not be equal to 0.

event_id – This field specifies an identifier of this event within the scope of scheme_id_uri and value. Events with the same value for scheme_id_uri, value, and event_id fields shall have same value for timescale, event_presentation_time_delta, event_duration, and event_data[].

`event_presentation_time_delta` – This field specifies presentation time of this event relative to the presentation time of the first access unit in this MPU. The relative value of this presentation time in seconds is equal to `event_presentation_time_delta` divided by `timescale`.

`event_duration` – This field specifies presentation duration of this event. The presentation duration of this event in seconds is equal to `event_duration` divided by `timeScale`.

`event_data` – The remaining data till end of this `'evti'` box specifies the data associated with this event. This field may be empty. The format of this field is defined by the owner of the scheme specified by `scheme_id_uri`.

One or more `'evti'` boxes may appear at the beginning of the MPU, after the `'ftyp'` box, but before the `'moov'` box, or they 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 4.3. The semantics of the fields in inband_event_descriptor() shall be as given immediately below the table.

Table 4.3 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.

4.2 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 4.1.2 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]). If the `Content-Type` header is provided in the HTTP response, it shall be `application/octet-stream` per [10].

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 4.4.

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 4.5.

For ROUTE/DASH-based services, the format of an Event delivered via HTTP or WebSocket server shall employ the same `'emsg'` box format described above together with information that associates it with a specific MPD Period as specified in Table 4.4. The semantics of the fields of this element shall as given immediately below the table.

Table 4.4 Syntax of the `'emsg'` Box Delivered over WebSocket or HTTP

Syntax	No. of Bits	Format
emsg_object() {		
mpd_id	var	uimsbf
mpd_id_terminator	8	0x00
period_id	var	uimsbf
period_id_terminator	8	0x00
segment_counter	32	uimsbf
emsg	var	
}		

mpd_id – The `MPD@id` attribute as defined in ISO/ IEC 23009-1 [5] of the MPD with which this Event is associated, in UTF-8 encoding. Note that this element may be of length zero (not present) if the MPD has no `MPD@id` attribute value assigned.

mpd_id_terminator – A null-terminator for the **mpd_id** field.

period_id – The `MPD.Period@id` attribute as defined in ISO/ IEC 23009-1 [5] of the MPD Period with which this Event is associated, in UTF-8 encoding. Note that this field may be of length zero (not present) if the MPD Period has no `MPD.Period@id` attribute value assigned.

period_id_terminator – A null-terminator for the **period_id** field.

segment_counter – An integer that identifies the segment of the MPD Period with which this Event is associated. The value of **segment_counter** shall be equal to the number of segments that precede the associated segment in the MPD Period identified by **period_id**. For example, if the Event is associated with the third segment of the MPD Period identified by **period_id**, then **segment_counter** will have the value 2.

emsg – An `'emsg'` box as defined in ISO/ IEC 23009-1 [5] containing an Event. Note that the `presentation_time_delta` of an Event specifies an offset relative to the start time of a segment, so the values of **mpd_id**, **period_id**, and **segment_counter** may be necessary to determine an unambiguous presentation time for the Event.

For MMTP/MPU-delivered services, the format of an Event delivered via HTTP or WebSocket server shall employ the same `'evti'` box format described above together with information that associates them with specific MMT Asset MPUs as specified in [7]. The semantics of the fields of this element shall be as given immediately below the table.

Table 4.5 Syntax of the `'evti'` Box Delivered over WebSocket or HTTP

Syntax	No. of Bits	Value	Format
evti_object() {			
asset_id_length	32	N	uimsbf
asset_id	N*8		uimsbf
mpu_sequence_number	32		uimsbf
evti	var		
}			

asset_id_length – The `asset_id_length` attribute as defined in ISO/ IEC 23008-1 [7].

asset_id – The `asset_id` attribute as defined in ISO/ IEC 23008-1 [7].

mpu_sequence_number – The `mpu_sequence_number` attribute as defined in ISO/ IEC 23008-1 [7].

evti – An `'evti'` box containing an Event as specified in section 4.1.2, where the `event_presentation_time_delta` is relative to the earliest access unit presentation time of the MPU referenced by **asset_id** and **mpu_sequence_number**.

4.3 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].

4.4 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 4. There are slight differences between ROUTE/DASH-based services and MMT-based services, but there is also significant commonality.

4.4.1 DASH-specific signaling Event Streams

The DASH-specific signaling Event Streams may be used for ROUTE/DASH-based Services or for MMT-based Services which includes broadband-delivered DASH segments. More details are described in Section 5.6 of the DASH-IF IOP for ATSC 3.0 [4].

4.4.2 Application-specific Event Streams

Application-specific Event Streams are defined by application developers. For ROUTE/DASH-based Services, constraints are described in Section 5.6.2 of the DASH-IF IOP for ATSC 3.0 [4]. For MMT-based Services, **`AEI`** and/or `'evti'` box can be used for Application-specific Event Streams signaling. The only constraints are that the combination of **`AEI.EventStream`**`@schemeIdUri`/**`AEI.EventStream`**`@value` attributes shall be globally unique, such as by the use of an **`AEI.EventStream`**`@schemeIdUri` attribute controlled by the application developer, and by proper management of the **`AEI.EventStream`**`@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.

4.4.3 ATSC-specific signaling Event Streams

ATSC-specific signaling Events are used to notify devices when unexpected updates to signaling metadata objects become available. For ROUTE/DASH-based Services, constraints and requirements are described in Section 5.6.2 and 5.6.3 of the DASH-IF IOP for ATSC 3.0 [4]. For MMT-based Services, the **`AEI.EventStream`**`@schemeIdUri` attribute shall be of form "`tag:atsc.org,2016:event`", and the **`AEI.EventStream`**`@value` attribute shall be "`stu`". If the dynamic event is used, the `event_data` field of `'evti'` box 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.

4.5 Event notification via WebSocket

4.5.1 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

4.5.2 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 [8] for receiving event notification (and optionally signaling object data notification) messages. Signaling object data includes such data as present in the HELD [1], 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
```

4.5.2.1 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.

4.5.2.2 EventNotify Subprotocol

The `EventNotify` subprotocol framing structure is shown in Figure 4.1. Table 4.6 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.

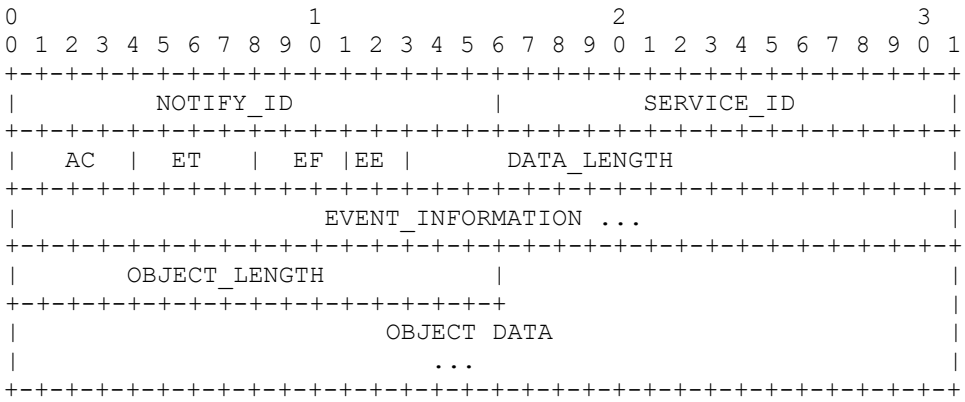

Figure 4.1 `EventNotify` subprotocol framing structure.

Table 4.6 `EventNotify` Subprotocol Framing Elements

Element	No. of Bits	Semantics
NOTIFY_ID	16	A notification identifier which uniquely identifies this event notification connection. 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. 0: Event notification from server to client. 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. 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. 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. 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. 5-15 : reserved.
ET	4	Indicates the type of event for which updated data is available. ET field value is interpreted as follows: 0 indicates ATSC ROUTE/DASH event. 1 indicates ATSC MMT event. 2-15 are reserved.
EF	3	Defines format of OBJECT_DATA. Following formats are defined. 0: Binary format used for OBJECT_DATA. 1: XML format used for OBJECT_DATA. 2: JSON format used for OBJECT_DATA.

		3-7: reserved.
EE	2	Defines encoding for OBJECT_DATA. Following encodings are defined. 0: No encoding. 1: GZIP encoding as per RFC 1952. 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. 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. 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 [5]. 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. 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. 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.

Annex A: Media Type Registrations

This Annex documents new media types registered by IANA at https://www.iana.org/assignments/media-types/media-types.xhtml#application. *Notice to editors: any changes to this Annex are subject to review by IETF and IANA as described in IETF BCP 13* [12].

A.1 AEI

Type name:

application

Subtype name:

mmt-aei+xml

Required parameters:

None.

Optional parameters:

None.

Encoding considerations:

binary

Same as for application/xml, except constrained to UTF-8. See IETF 7303 [13], Section 9.1.

Security considerations:

This media type inherits the issues common to all XML media types - see RFC 7303 [13] Section 10. 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. This type does not employ executable content, but since it is explicitly extensible then executable content could appear in an extension. This media type does not provide any confidentiality protection and instead relies on the transport protocol that carries it to provide such security, if needed.

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 4.1.2. This specification and XML schema for

the content are available at www.atsc.org/standards (the schema(s) are provided in a zip file).

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

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 –