CHANGE REQUEST

DASH-IF IOP **CR** **rev** **-** **Current version:** V4.3

Status:		Draft		Internal Review	X	Community Review		Agreed

Title:	WebIDL DASH Event Metadata API
Source:	DASH-IF IOP Event TF
Supporting Companies:	Qualcomm Incorporated, LG Electronics, Sony, Ericsson, Nomor Research
Category:	**A** ***Date:*** 2019-08-06

Use one of the following categories:
C *(correction)*
A *(addition of feature)*
B *(editorial modification)*

Reason for change:	For an API description suitable across platforms that corresponds to the proper usage of Events and Timed Metadata distributed in DASH Media Presentations, APIs pertaining to subscription and notification delivery are beneficially defined between the DASH client and the application consuming the Events.
Summary of change:	Addition of a client processing model for Events
Consequences if not approved:	Inconsistent implementations

Sections affected:	New section X
Other comments:	This document contains several notes. Feedback during community review is welcome specifically on these topics.

Disclaimer:	This document is not yet final. It is provided for public review until the deadline mentioned below. If you have comments on the document, please submit comments by one of the following means: - at the github repository https://github.com/Dash-Industry-Forum/Events/issues, or - dashif+iop@groupspaces.com with a subject tag [Events] Please add a detailed description of the problem and the comment. Based on the received comments a final document will be published latest by the expected publication date below, integrated in a new version of DASH-IF IOP if the following additional criteria are fulfilled: - All comments from community review are addressed - The relevant aspects for the Conformance Software are provided - Verified IOP test vectors are provided
Commenting Deadline:	August 31st, 2019
Expected Publication:	September 30th, 2019

1. Abstract

This document specifies an API that a user agent or DASH client can expose for application access to DASH events. This builds upon Media Source Extensions. The API correspond to partial interfaces with respect to the `DASHEvent` event target. It is meant to cover both in-band (`'emsg'`) and MPD-carriage of events, as well as sparse timed metadata tracks.

> Note: The document implements the processing model as defined in the DASH-IF document on "DASH Player's Application Events and Timed Metadata Processing Models and APIs". The community review version is available here: dashif-documents.azurewebsites.net/Events/master/event.html. The combination of the two documents and cross-referencing will be dealt in the final version of this document.

2. DASHEvent Interface

2.1 Definition

```
[Constructor(SourceBuffer source)]
interface DASHEvent : EventTarget {
    readonly        attribute EventData             eventData;
                    attribute EventHandler          ondashevent;
    Promise         setEvents(Eventlist eventList);
};
```

2.2 Attributes

`eventData` of type `EventData`, readonly

> When an event is encountered, the DASH client *MUST* extract the event data, and *MUST* initialize the object's `eventData` attribute to a string representation of the event data.

`ondashevent` of type EventHandler

> This event handler is invoked when a new DASH event arrives.

`setEvents()` of type Promise

> This promise must include an `eventList` argument that enumerates all events in which the application is interested.

3. EventData Interface

3.1 Definition

```
interface EventData {
    readonly        attribute DOMString schemeIdURI;
    readonly        attribute DOMString value;
    readonly        attribute DOMTimeStamp? presentationTime;
    readonly        attribute unsigned long? duration;
    readonly        attribute unsigned long? id;
    readonly        attribute ByteString messageData;
};
```

3.2 Attributes

`schemeIdURI` of type DOMString, readonly

> The `schemeIDURI` attribute *MUST* return a URI that identifies the DASH event scheme.

`value` of type DOMString, readonly

> The `value` attribute *MUST* return the value for the event stream element. The value semantics are defined by the owners of the scheme identified in the `schemeIdUri` attribute.

`presentationTime` of type DOMTimeStamp, readonly

> The `presentationTime` attribute *MUST* return a value corresponding to the exact moment in the media presentation timeline that the event becomes active. If this attribute is not present then its value shall be set to NULL and the event is assumed to be active immediately.

`duration` of type int, readonly

> The `duration` attribute *MUST* return the time for which the event is in effect starting from presentationTime. The value of the duration is in milliseconds. If this attribute is not present then its value must be set to the maximum value (4294967295) and the event *MUST* be persisted until another DASH event is received.

`id` of type int, readonly

> The `id` attribute *MUST* return an identifying value for this event. If this value is not present then its value must be set to NULL.

`messageData` of type ByteString, readonly

> The `messageData` attribute *MUST* return the event message data payload.

4. EventList Interface

4.1 Definition

```
dictionary EventList {
    DOMString[]             desiredSchemeIdURI;
    optional DOMString[]    value;
    optional boolean[]      dispatchMode;
};
```

`EventList` contains one or more valid event scheme URI's along with associated parameters.

4.2 Members

`desiredSchemeIdURI` of type `DOMString[]`

> `desiredSchemeIdURI` is an array of valid DASH event scheme URI's. If the `desiredSchemeURI` array is set to NULL, then all events will be sent to the handler.

`value` of type `DOMString[]`

> `value` is an array of valid values for events. If the `value` array has only one member, then that value will be applied to all scheme URI's. If more than one member is present in the array then those values will be matched to each member in `desiredSchemeIdURI` in order. A `value` array with more members than the `desiredSchemeIdURI` array should be rejected if the `desiredSchemeIdURI` array is non-NULL. In the case of a non-NULL `desiredSchemeIdURI` array, members of the `value` array can be set to NULL if all values are acceptable for the associated Scheme URI. If the `desiredSchemeIdURI` array is NULL, then the `value` array can have only one member.

`dispatchMode` of type `boolean[]`

`dispathMode` is an array of dispatch mode settings for events (true indicating a dispatch on receipt of the event, false representing a dispatch on the start of the event) . If the `dispatchMode` array has only one member, then that dispatch mode will be applied to all scheme URI's. If more than one member is present in the array then those values will be matched to each member in `desiredSchemeIdURI` in order. A `dispatchMode` array with more members than the `desiredSchemeIdURI` array should be rejected if the `desiredSchemeIdURI` array is non-NULL. In the case of a non-NULL `desiredSchemeIdURI` array, members of the `dispatchMode` array can be set to NULL if both dispatch modes are acceptable for the associated Scheme URI. If the `desiredSchemeIdURI` array is NULL, then the `dispatchMode` array can have only one member.

5. Example

```
<html>
<body>

<script>
  function onSourceOpen(videoTag, e) {
    var mediaSource = e.target;
    if (mediaSource.sourceBuffers.length > 0)
        return;
    try {
        dashevent = new DashEvent(mediaSource);
        dashevent.setEvents(["schemeURI1","schemeURI2"]).then(
            {
            console.log('Desired event list set');
            }
        }
        catche (e)
            {
            console.error('Failed to create Dash event handler due to:  ' + e);
            return;
            }
    dashevent.ondashevent = dashEventHandler;
    function dashEventHandler(event){
        }
    }
</script>
<video id="v" autoplay> </video>
<script>
  var video = document.getElementById('v');
  var mediaSource = new MediaSource();
  mediaSource.addEventListener('sourceopen', onSourceOpen.bind(this, video));
  video.src = window.URL.createObjectURL(mediaSource);
</script>
</body>
</html>
```