14,

ATSC Standard: A/344:2019 Amendment No. 8, Signaling Data

Doc. A/344:2019 Amend. No. 8
7 February 2020

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

The Advanced Television Systems Committee, Inc., is an international, non-profit organization developing voluntary standards and recommended practices for digital television. ATSC member organizations represent the broadcast, broadcast equipment, motion picture, consumer electronics, computer, cable, satellite, and semiconductor industries. ATSC also develops digital television implementation strategies and supports educational activities on ATSC standards. ATSC was formed in 1983 by the member organizations of the Joint Committee on Inter-society Coordination (JCIC): the Electronic Industries Association (EIA), the Institute of Electrical and Electronic Engineers (IEEE), the National Association of Broadcasters (NAB), the National Cable Telecommunications Association (NCTA), and the Society of Motion Picture and Television Engineers (SMPTE). For more information visit www.atsc.org.

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

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

Revision History

Version	Date
Amendment approved	7 February 2020

ATSC Standard:
A/344:2019 Amendment No. 8, Signaling Data

1. OVERVIEW

1.1 Definition

An Amendment is generated to document an enhancement, an addition or a deletion of functionality to previously agreed technical provisions in an existing ATSC document. Amendments shall be published as attachments to the original ATSC document. Distribution by ATSC of existing documents shall include any approved Amendments.

1.2 Scope

This document adds query and notification APIs to access the LLS and SLS signaling tables.

1.3 Rationale for Changes

Apps have a need to obtain basic service information (see A/331), specifically including:

- serviceId
- majorChannelNo and minorChannelNo
- protected and hidden
- serviceCategory and broadbandAccessRequired

In addition to anticipate future needs to access signaling, it is desirable to provide access to all LLS and SLS table information.

Consider deprecating the resulting overlapping APIs.

1.4 Compatibility Considerations

This amendment is backwards compatible since the changes are the addition of new APIs. (Existing) receiver implementations need not support the new APIs. Additionally, when supported by a Receiver, existing Broadcaster Applications are expected to ignore items that they do not understand per the provisions of Section 9.1: "*The Broadcaster Application is expected to gracefully ignore unknown keys and unknown values for existing keys, including unknown enumeration values.*"

The existing APIs, to be deprecated, for now remain in force and thus have no impact at this time.

2. LIST OF CHANGES

Change instructions are given below in *italics*. Unless otherwise noted, inserted text, tables, and drawings are shown in red (due to text coloring in base document); deletions of existing text are shown in ~~red strikeout~~. The text "[ref]" indicates that a cross reference to a cited referenced document should be inserted.

A/344 maintains a "revision log" of its included APIs from revision to revision by listing the changes in Table 9.1. In addition, each revision includes an Annex that captures the API from the previous edition in unchanged form. By maintaining the previous API definition in the document, implementers may look at the history of each API. When this amendment is finally rolled into the main revision document, Table 9.1 will need to be updated and the original text of the API modified below may be copied into the Annex for the revision.

Add a new section 9.2.x as follows. Note that this section is specified in the colors intended instead of the normal red designated for changes.

9.2.x Query Signaling API

The Broadcaster Application may wish to access the various signaling metadata structures from the current broadcast. The Query Signaling API returns a list of signaling tables that the Broadcaster Application can use to extract details not otherwise available such as major and minor channel numbers.

The Query Signaling API shall be defined as follows:

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

params: A JSON object consisting of a key named `objectNames` with a list of enumerated values.

params JSON Schema:

```
{
   "type": "object",
   "properties": {
      "nameList": {
         "type": "array",
         "items": { "type": "string" }
      },
      "required": ["nameList"]
   }
}
```

`nameList` – An array of object names as described below in `names`. If absent, no tables or Metadata Object Types are returned.

`names` – This field shall be set to an integrated list of `LLS_table_id` values and signaling metadata `object_type` values as enumerated in A/331, Table 6.17, "Metadata Object Types". The `LLS_table_id` values shall be encoded as a decimal integer (i.e. not hex). When the `names` field is omitted, this request shall return all of the supported current LLS tables and all of the supported current Metadata Object Types. Note: Some Metadata Object Types are transport dependent (ROUTE versus MMT) and might not be available on a given Receiver model. A request to return the LLS SignedMultiTable, ALL, AEI, EMSG, and EVTI is not expected to return any LLS tables or Metadata Object Types; that is, no such LLS tables and Metadata Object Types as well as any unsupported ones would be returned. Note that LLS tables can be delivered via the SignedMultiTable.

Response:

result: a JSON object containing a list of tables as text strings as defined below.

result JSON Schema:

```
{
  "type": "object",
   "properties": {
      "objectList": {
         "type": "array",
         "items": {
            "name": { "type": "string" },
            "group": { "type": "integer" },
            "version": { "type": "integer" },
            "table": { "type": "string", "format": "xml" },
            "required": ["name", "version", "table"]
         }
      },
      "required": ["objectList"]}
   }
}
```

`name` – See `names` above.

`version` – This is the version of the XML signaling document. For LLS, it shall be set to the value of `LLS_table_version`. For Metadata Object Types, it shall be set to `metadataEnvelope@version`.

`group` – For LLS tables, this is required and shall be the value of `LLS_group_id`. For Metadata Object Types this shall be omitted.

`table` – This string shall contain a single object of a type matching `name`, encoded in UTF-8. For objects that are not strings, they shall first be encoded as Base64. The XML documents shall be uncompressed. For both LLS and SLS documents, only the XML document itself is returned. Related packaging including signatures, binary LLS fields (e.g. `group_count_minus1`), MIME separators, etc. shall be removed.

For example, the application makes a query for the SLT and MPD as follows:

```
--> {
    "jsonrpc": "2.0",
    "method": "org.atsc.query.signaling",
    "params": {
        "objectList": ["1", "MPD"]
    },
    "id": 913
}
```

The Receiver might respond:

```
<-- {
    "jsonrpc": "2.0",
    "result": {
        "objectList": [
          { "name": "1",
            "group": "1"
            "version": "23",
            "table": "<SLT …… </SLT>" },
          { "name": "MPD",
            "version": "65",
            "table": "<MPD …… </MPD>" }        ]
    },
    "id": 913
}
```

Add a new section 9.3.x as follows. Note that this section is specified in the colors intended instead of the normal red designated for changes.

9.3.x Signaling Data Change Notification API

The Signaling Data Change Notification API shall be issued by the Receiver to the currently executing Broadcaster Application if a new version of any LLS table or SLS fragment is received since either the Broadcaster Application subscribed to receive such notifications via the API specified in Section [9.7.5] or a version was previously notified. The Broadcaster Application may respond to the notification of a change to the signaling data by using the Query Signaling Data API specified in Section 9.3.x to fetch a new copy.

The Signaling Data Change Notification API shall be defined as follows:

method: "`org.atsc.notify`"

params: A JSON object consisting of a key named `msgType` with value "`signalingData`"

params JSON Schema:

```
{
  "type": "object",
  "properties": {
    "msgType": {"type": "string", "enum": ["signalingData"]},
  },
  "required": ["msgType"]
}
```

The following is an example notification:

```
<-- {
    "jsonrpc": "2.0",
    "method": "org.atsc.notify",
    "params": {
        "msgType": "signalingData",
    }
}
```

Revise Table 9.3 as follows:

Table 9.3 Subscription Parameter List

Notification APIs	**Reference**	**msgType**
All Notification APIs	-	All
Signaling Data APIs	9.3.x	signalingData
Rating Change Notification API	9.3.1	ratingChange
Rating Block Change Notification API	9.3.2	ratingBlock
Service Change Notification API	9.3.3	serviceChange
Caption State Change Notification API	9.3.4	captionState
Language Preference Change Notification API	9.3.5	languagePref
Caption Display Preferences Change Notification API	9.3.6	CCDisplayPref
Audio Accessibility Preference Change Notification API	9.3.7	audioAccessPref
MPD Change Notification API	9.3.8	MPDChange
Alerting Change Notification API	9.3.9	alertingChange
Content Change Notification API	9.3.10	contentChange
Service Guide Change Notification API	9.3.11	serviceGuideChange
Content Recovery State Change Notification API	9.10.4	contentRecoveryStateChange
Display Override Change Notification API	9.10.5	displayOverrideChange
Recovered Component Info Change Notification API	9.10.6	recoveredComponentInfoChange
RMP Media Time Change Notification API	9.14.5	rmpMediaTimeChange
RMP Playback State Change Notification API	9.14.6	rmpPlaybackStateChange
RMP Playback Rate Change Notification API	9.14.7	rmpPlaybackRateChange
DRM Notification API	9.15.1	DRM
XLink Resolution Notification API	9.16.1	xlinkResolution

Deprecate the following:

- Integrated Subscribe / Unsubscribe API for Notification
  - `msgType="MPDChange"`
- MPD Change Notification API (9.3.8)
- Query MPD URL API (9.2.7)

Note that deprecated interfaces need to be moved to a new deprecated Annex for A/344:2019.

– End of Document –