ATSC Standard: A/344:2019 Amendment No. 7, CacheRequest API

Doc. A/344:2019 Amend. No. 7
25 December 2019

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

Revision History

Version	Date
Amendment approved	25 December 2019

ATSC Standard: A/344:2019 Amendment No. 7, CacheRequest API

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 explains the inconsistent method names of the Cache Request APIs in Section 9.4 so that implementors are not confused about whether the names contain typos or not.

1.3 Rationale for Changes

The existing Cache Request APIs defined in Section 9.4 do not have names consistent with other API calls throughout the document. In brief, both commands start with a capital "C" when, to be consistent, they should start with lower case "c". This inconsistency could lead to various implementation errors where some implementors recognize the inconsistency and "fix it" while others follow the current standard verbatim.

Text is proposed that explains that the inconsistency is known and that implementors should use the method names verbatim instead of assuming a problem and being tempted to "fix" the offending letter.

1.4 Compatibility Considerations

This amendment is backwards compatible since the changes are entirely editorial and do not change any APIs. Note that previous implementations that did not follow the standard verbatim will not be compliant.

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 base standard text coloring); 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 which 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.

Change section 9.4.1 as follows:

9.4.1 Cache Request API

The Broadcaster Application can use the Cache Request API to request that the Receiver download one or more indicated files. The Broadcaster Application might request to download ad content via broadband before the time of an ad replacement to avoid playback problems that might occur due to network congestion if the ad were to be streamed in real time. Note that retrieval and storage of broadcast- delivered NRT personalized content may be managed by the Filter Codes APIs (see Section [ref 9.11]). The Cache Request API includes Filter Codes as well to aid in the Receiver's management of objects in the Application Context Cache.

The Receiver's response to the Cache Request API indicates whether or not the indicated files are already present in the Application Context Cache, thus the API may also be used to check whether or not the one or more indicated files are present in the Application Context Cache. The status check function works for files that might have arrived by either the broadcast or the broadband delivery path.

As stated in Section [ref 6.2], storage capability and management of the Application Context Cache are receiver-specific, so that files requested via this API might or might not be stored, depending on the status of the Application Context Cache. However, the Broadcaster Application can use the Query Cache Usage API defined in Section [ref 9.5] to check how much storage quota of Application Context Cache is assigned for the Application Context ID. The Mark Unused API defined in Section [ref 9.9] can be used to indicate to the Application Context Cache system that cached file(s) are unused. If the currently- executing Broadcaster Application is terminated, the Receiver may cancel all in-progress file retrieval processes and release all cached files requested by this API.

Note that the method name, `"org.atsc.CacheRequest"`, starts with a capital "C" inconsistent with the method naming in other parts of this standard. The reader is cautioned to use the method name verbatim to avoid issues.

The Cache Request API shall be defined as follows;

method: `"org.atsc.CacheRequest"`

params: A JSON object containing the parameters of the method;

params JSON Schema:

…

Note that the examples in Section 9.4.1 need to be changed to match the method name (upper case "C") to avoid confusion as follows:

To accomplish the download, the Broadcaster Application could issue the following API:

```
--> {
    "jsonrpc": "2.0",
    "method": "org.atsc.~~e~~CacheRequest",
    "params": {
      "sourceURL": "https://foo.com/service1/",
      "targetURL": "images/",
      "URLs":["A/big-image1.png","B/big-image2.png","C/big-image3.png"],
      "filters": [1007, 1009]
    },
    "id": 37
  }
```

In this example, the first PNG file is fetched using the URL `https://foo.com/service1/A/big-image1.png` and placed into the Application Context Cache at a subdirectory `images/A/big-image1.png`.

Upon successfully beginning the retrieval process, if the files had not been retrieved previously, the Receiver would respond with:

```
<-- {
    "jsonrpc": "2.0",
    "result": {"cached": false},
    "id": 37
  }
```

Note that `cached` is "false", indicating that these files are not already present. If all the files had already been present in the Application Context Cache and none had expired, `cached` would have returned "true", otherwise the Receiver would begin to re-download the files. If the Broadcaster Application wishes later to check to see whether or not the first two of these files have been successfully downloaded, it could issue the following API to the Receiver:

```
--> {
    "jsonrpc": "2.0",
    "method": "org.atsc.~~e~~CacheRequest",
    "params" {
      "targetURL": "images/",
      "URLs":["A/big-image1.png", "B/big-image2.png"]
    },
    "id": 38
  }
```

If both of the indicated files are present and not expired, the Receiver may respond with:

```
<-- {
    "jsonrpc": "2.0",
    "result": {"cached": true},
    "id": 38
}
```

Similarly, change Section 9.4.2 as follows:

9.4.2 Cache Request DASH API

The Cache Request DASH API may be used by the currently- executing Broadcaster Application to indicate to the Receiver that certain files should be retrieved via broadband and stored in the Application Context Cache. Instead of listing each URL individually, using this API, files are specified either in an MPEG DASH Period XML fragment or in a complete DASH MPD. If a complete DASH MPD is specified, the MPD file and the MPEG DASH segments specified in the MPD file shall be retrieved via broadband and stored. The URL of each MPEG DASH segment file shall be generated according to the MPEG DASH specification [ref DASH]. In response to the XLink Resolution API, the Broadcaster Application can provide the same DASH Period XML fragment.

The Cache Request DASH API may also be used to check whether or not the files indicated in the DASH Period or MPD are present in the Application Context Cache and not expired. The status check function works for files that might have arrived by either the broadcast or the broadband delivery path.

Note that the method name, `"org.atsc.CacheRequestDASH"`, starts with a capital "C" inconsistent with the method naming in other parts of this standard. The reader is cautioned to use the method name verbatim to avoid issues.

The Cache Request DASH API shall be defined as follows;

method: `"org.atsc.CacheRequestDASH"`

params: A JSON object containing the parameters of the method;

params JSON Schema:

Furthermore, the existing examples in Section 9.4.2 need not be changed because they are already consistent with the method name specified.

– End of Document –