
# ATSC Standard: A/344:2019 Amendment No. 2, JSON RPC Cancel Request API Addition

Doc. A/344:2019 Amend. No. 2
16 December 2019

**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 | 16 December 2019 |

# ATSC Standard: A/344:2019 Amendment No. 2, JSON RPC Cancel Request API Addition

## 1. OVERVIEW

### 1.1 Definition

The published version of A/344 and the included JSON RPC 2.0 specification does not consider how long a JSON RPC request / response transaction may take and provides no way to manage that transaction. This amendment describes the addition of a cancel request operation that may be used to cancel outstanding JSON RPC requests. An error response will be generated for the outstanding request when the cancel request is successful.

### 1.2 Scope

This document provides an addition to Section 8.3 in A/344:2019 which, in turn, applies to all request / response API operations defined in Section 9.

### 1.3 Rationale for Changes

The changes described in this document give a Broadcaster Application more control over request / response API transactions.

The cancel request provides a way for the Broadcaster Application to limit how long a transaction takes. Some request / response transactions can take a significant amount of time between when the request is made, and when the transaction completes resulting in the response being issued by the Receiver. The Broadcaster Application may be aware of how long a transaction should take and could desire to terminate that transaction if it is not accomplished in the time given.

There may be circumstances where the Broadcaster Application may wish to terminate the current operation and move to another function entirely. This change could be requested by the user through some interface or could be caused by other external factors or events. In the present system, the Broadcaster Application must keep track of outstanding requests and handle the responses whenever they may occur. The cancel operation allows all outstanding requests to be terminated efficiently.

### 1.4 Compatibility Considerations

The changes described in this document are backward-compatible since the cancel request is in addition to the current A/344 APIs. All APIs will continue to operate as before if the cancel request is not used. The cancel command is optional and can safely be ignored if Broadcaster Applications do not wish to use it. Existing Broadcaster Applications would operate exactly the same without modification if these API changes were added. Note that issuing the cancel request to receivers that do not support the Cancel Request API is expected to result in the standard error response to unknown requests, that is, error code -32600, Invalid Request, as defined in the JSON-RPC 2.0 Specification duplicated in Annex D.

## 2. LIST OF CHANGES

Change instructions are given below in *italics*. Unless otherwise noted, inserted text, tables, and drawings are shown in blue; 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.

The following indicate changes to Section 8.3 of A/344:2019.

## 8.3 Data Binding

Once the connection is established to the Receiver WebSocket command and control Server, messages can be sent and received. However, since the WebSocket interface is just a plain bidirectional interface with no structure other than message framing, a message format needs to be defined. This section defines the basic formatting of messages, and the following section defines the specific messages that are supported. The message syntax used in this document is defined in the JSON Schema specification [33].

The WebSocket interface for command and control shall be the JSON-RPC 2.0 Specification in Annex D. JSON-RPC provides RPC (remote procedure call) style messaging, including unidirectional notifications and well-defined error handling using the JavaScript Object Notation (JSON) data structure [18].

The data is always sent as a UTF-8 stringified JSON object. The Receiver shall parse the JSON object and route the method to the right handler for further processing. Several types of data messages are defined for the command and control WebSocket interface:

Request message – used to request information or initiate an action

Synchronous response – a definitive answer to a request provided immediately

Asynchronous response – a definitive answer to the request provided asynchronously

Error response – a definitive error to the request provided

Notification – unidirectional notification, no synchronous or asynchronous response is expected

The other three WebSocket interfaces are used for delivery of binary data from the Receiver to the Broadcaster Application.

The notation used to describe the flow of data in this specification shall be as follows:

```
--> data sent to Receiver
<-- data sent to Broadcaster Application
```

Note: The interface is bidirectional, so requests, responses and notifications can be initiated by either the Receiver or the Broadcaster Application.

Request/response example:

```
--> {"jsonrpc": "2.0",
    "method": "exampleMethod1",
    "params": 1,
    "id": 1
}

<-- {
    "jsonrpc": "2.0",
    "result": 1,
    "id": 1
}
```

Notification example:

```
--> {
    "jsonrpc": "2.0",
    "method": "update",
    "params": [1,2,3,4,5]
}
```

Note that the lack of an ‘id’ term indicates that no response is expected in the case of a notification.

Error example:

```
--> {
    "jsonrpc": "2.0",
    "method": "faultyMethod",
    "params": 1,
    "id": 6
}

<-- {
    "jsonrpc": "2.0",
    "error": {"code": -32601, "message": "Method not found"},
    "id": 6
}
```

### 8.3.1 Cancel Request Command

The Cancel Request Command may be used to terminate a selected number or all outstanding JSON RPC requests. The request message supplies a list of IDs corresponding to the request IDs. The response message lists the requests that have been terminated. If no requests matching the requested IDs can be found, an error response is provided. If a request is successfully canceled, the Receiver shall issue a response to that request with an error code indicated that the request was canceled. A cancel request shall not be used to cancel a previous cancel request.

The Cancel Request Command shall be defined as follows:

method: "`cancel`"

params: A JSON object consisting of a key named `requestIDs` containing a list of request IDs to be canceled.

params JSON Schema:

```
{
  "type": "object",
  "properties": {
    "requestIDs": {
      "type": "array",
      "items": {"type": "integer"}
    }
  }
}
```

`requestIDs` – This optional parameter shall contain one or more IDs of outstanding requests. If the `requestIDs` list is not supplied, then all outstanding requests shall be canceled.

Response:

result: a JSON object containing the list, `cancelList`, of objects, each object containing the canceled request ID, `requestID`, the disposition of that cancel request, `disposition`, and a description of the disposition, `description`. The Receiver shall accumulate the canceled request IDs so that a single response can be supplied. Note that each request receives a separate error response indicating that the request was canceled.

result JSON Schema:

```
{
   "type": "object",
   "properties": {
      "cancelList": {"type": "array", "items": { "type": "object",
         "properties": {
            "requestID": {"type": "integer"}
            "disposition": {"type": "string", "enum": ["CANCELED", "UNKNOWN",
"FAILED"]},
            "description": {"type": "string"},
         },
         "required": ["requestID", "disposition"]
      }},
   },
   "required": ["cancelList"]
}
```

`cancelList` – This required property shall provide a list of objects in response to the cancel request. The list shall have the same number of objects as the number of request IDs in the cancel command or the number of outstanding requests if no request IDs were supplied requesting that all outstanding requests be canceled.

`requestID` – This required property shall contain one of the request IDs provided in the cancel command or the `requestID` of a canceled request if all requests were to be canceled.

`disposition` – This required property shall contain one of the following values:

- `CANCELED` – Shall indicate that the request corresponding to the object's `requestID` was successfully canceled.
- `UNKNOWN` – Shall indicate that the request corresponding to the object's `requestID` was not found. The request was not canceled because the Receiver could not identify an outstanding request corresponding to the given `requestID`, or because the request with that ID value was completed before the cancel request was processed.
- `FAILED` – Shall indicate that the request corresponding to the object's `requestID` could not be canceled. In this case, the Receiver found the specified request but could not successfully cancel the request.

`description` – This optional property shall contain a description of the cancel operation.

For example, the Broadcaster Application makes a query request with ID '12' and, due to a user action removing the need for the query, the Broadcaster Application cancels the request with the following command:

```
--> {
    "jsonrpc": "2.0",
    "method": "cancel",
    "params": {
        "requestIDs": [12]
    },
    "id": 913
}
```

The Receiver might respond to the cancel request as follows:

```
<-- {
    "jsonrpc": "2.0",
    "result": {
        "cancelList": [
          { "requestID": 12,
            "disposition": "CANCELED",
            "description": "Request canceled successfully"  }
        ]
    },
    "id": 913
}
```

The Receiver might also issue the following response to query request with ID 12:

```
<-- {
    "jsonrpc": "2.0",
    "error": {"code": -20, "message": "Request Canceled"},
    "id": 12
}
```

As a further example, the Broadcaster Application may wish to cancel all outstanding requests of a function because the user has switched modes of operation. In this case, the Broadcaster Application attempts to cancel the corresponding outstanding requests as follows:

```
--> {
    "jsonrpc": "2.0",
    "method": "cancel",
    "params": {
        "requestIDs": [42, 216, 922]
    },
    "id": 226
}
```

The Receiver might respond to the cancel request as follows:

```
<-- {
    "jsonrpc": "2.0",
    "result": {
        "cancelList": [
          { "requestID": 42,
            "disposition": "UNKNOWN",
            "description": "Request is not currently pending"  }
          { "requestID": 216,
            "disposition": "CANCELED",
            "description": "Request canceled successfully"  }
          { "requestID": 922,
            "disposition": "CANCELED",
            "description": "Request canceled successfully"  }        ]
    },
    "id": 226
}
```

The Receiver issues error responses to query requests with IDs 216 and 922. Note that the `UNKNOWN` disposition for `requestID` 42 may not be an error since the request could have completed just as the cancel command was issued.

As a further example, the Broadcaster Application wishes to shut down all operations and terminate all outstanding requests. In this case, the Broadcaster Application attempts to cancel all outstanding requests as follows:

```
--> {
    "jsonrpc": "2.0",
    "method": "cancel",
    "id": 226
}
```

The Receiver might respond to the cancel request as follows:

```
<-- {
    "jsonrpc": "2.0",
    "result": {
        "cancelList": [
          { "requestID": 324,
            "disposition": "CANCELED",
            "description": "Request canceled successfully"  }
          { "requestID": 167,
            "disposition": "CANCELED",
            "description": "Request canceled successfully"  }        ]
    },
    "id": 226
}
```

The Receiver also issues error responses to query requests with IDs 324 and 167. The Broadcaster Application can assume that all outstanding requests have been canceled.

### 8.3.2 Error handling

JSON-RPC 2.0 defines a set of reserved error codes. See the table in Annex D Section 5.1.

ATSC-defined error codes from the Receiver shall be as defined in Table 2.1.

**Table 2.1** JSON-RPC ATSC Error Codes

| Code | Message | Meaning |
|---|---|---|
| -1 | Unauthorized | Request cannot be honored due to domain restrictions. |
| -2 | Not enough resources | No resources available to honor the request. |
| -3 | System in standby | System is in standby. Request cannot be honored. |
| -4 | Content not found | Requested content cannot be found. For example, invalid URL. |
| -5 | No broadband connection | No broadband connection available to honor the request. |
| -6 | Service not found | The requested Service cannot be located. |
| -7 | Service not authorized | The requested Service was acquired but is not authorized for viewing due to conditional access restrictions. |
| -8 | Video scaling/position failed | The request to scale and/or position the video did not succeed. |
| -9 | XLink cannot be resolved | The request to resolve an XLink has failed. |
| -10 | Track cannot be selected | The media track identified in the Media Track Selection API cannot be found or selected. |
| -11 | The indicated MPD cannot be accessed | In response to the Set RMP URL API, the MPD referenced in the URL provided cannot be accessed. |
| -12 | The content cannot be played | In response to the Set RMP URL API, the requested content cannot be played. |
| -13 | The requested MPD Anchor cannot be reached | In response to the Set RMP URL API, the MPD Anchor indicated cannot be reached (e.g. beyond the end of the file). |
| -14 | Unsupported Content Protection System | The specified content protection system is not supported by the Receiver. |
| -15 | Illegal URL Format | The URL format specified in `sourceURL` or `targetURL` of the request is illegal. |
| -16 | Illegal URL Format | The URL format specified in one or more URLs in the requested list is illegal. |
| -17 | Malformed DASH Period | The format of the MPEG DASH fragment specified in the Period is illegal. |
| -18 | MPD not found | The referenced MPD file cannot be found. |
| -19 | The synchronization specified by rmpSyncTime cannot be achieved | In response to the Set RMP URL API with rmpSyncTime, the synchronization indicated by rmpSyncTime cannot be achieved. |
| -20 | Request Canceled | The Broadcaster Application issued the cancel request command (Section 8.3.1) with the ID corresponding to this request. |

– End of Document –