Session Border Controllers (SBC)

Multi-Service Business Routers (MSBR)

VoIP Analog & Digital Media Gateways

REST API

For Mediant Devices

HD VoIP Sounds Better

Version 1.0

August 2017

Document # LTRT-41761

AudioCodes

Table of Contents

This page is intentionally left blank.

Notice

Information contained in this document is believed to be accurate and reliable at the time of printing. However, due to ongoing product improvements and revisions, AudioCodes cannot guarantee accuracy of printed material after the Date Published nor can it accept responsibility for errors or omissions. Before consulting this document, check the corresponding Release Notes regarding feature preconditions and/or specific support in this release. In cases where there are discrepancies between this document and the Release Notes, the information in the Release Notes supersedes that in this document. Updates to this document and other documents as well as software files can be downloaded by registered customers at http://www.audiocodes.com/downloads.

Date Published: August-06-2017

Trademarks

WEEE EU Directive

Pursuant to the WEEE EU Directive, electronic and electrical waste must not be disposed of with unsorted waste. Please contact your local recycling authority for disposal of this product.

Customer Support

Customer technical support and services are provided by AudioCodes or by an authorized AudioCodes Service Partner. For more information on how to buy technical support for AudioCodes products and for contact information, please visit our Web site at www.audiocodes.com/support.

Abbreviations and Terminology

Each abbreviation, unless widely used, is spelled out in full when first used. Device refers to SBC and gateway.

Documentation Feedback

AudioCodes continually strives to produce high quality documentation. If you have any comments (suggestions or errors) regarding this document, please fill out the Documentation Feedback form on our Web site at http://www.audiocodes.com/downloads.

This page is intentionally left blank.

1 Overview

The REST API is designed for developers who wish to programmatically integrate the Mediant Gateway or SBC device into their solution and for administrators who wish to perform management and configuration tasks via automation scripts.

The REST API provides access to the resources via pre-defined URL paths. Each resource represents specific device configuration element, state object or maintenance action.

The REST API uses standard HTTP/1.1 protocol. For enhanced security it is recommended to secure the traffic via the use of HTTPS transport layer.

Standard HTTP methods – GET, PUT, POST and DELETE – are used to read the resource's state and to create/update/delete the resources (wherever applicable). Resource state is described in JSON format and included in the HTTP request or response bodies.

This page is intentionally left blank.

2 Authentication and Session Establishment

The REST API is accessible via HTTP/HTTPS protocol at `/api/v1` prefix.

Example

```
GET http://10.4.219.62/api/v1/status
```

```
HTTP/1.1 200 OK
Content-Type: application/json
{
    "localTimeStamp": "2010-01-17T17:29:15.000Z",
    "ipAddress": "10.4.219.62",
    "subnetMask": "255.255.0.0",
    "defaultGateway": "10.4.0.1",
    "productType": "Mediant SW",
    "versionID": "7.00A.018.005",
    "protocolType": "SIP",
    "operationalState": "UNLOCKED",
    "highAvailability": "Stand Alone"
}
```

Each REST request must be authenticated using HTTP Basic Authentication. Provided credentials should correspond to a valid device user with the Security Administrator privilege level.

Note: It is strongly recommended to use the HTTPS transport layer when accessing the REST API to mitigate security risks.

This page is intentionally left blank.

3 Top-Level Folder

The `/api` URL serves as a root folder for accessing the REST API.

URL

```
/api
```

HTTP Method

```
GET
```

HTTP Response

```
200 OK
```

Example

```
GET /api

HTTP/1.1 200 OK
Content-Type: application/json
{
    "versions": [
        {
            "id": "v1",
            "status": "stable",
            "url": "/api/v1"
        }
    ]
}
```

This page is intentionally left blank.

4 Navigation Tree

The `/api/v1` URL displays the complete navigation tree that is supported by the REST API. This tree is displayed below:

```
/api/v1
       /actions
          /reset                // reset the device
          /saveConfiguration    // save configuration to NVRAM
       /files                   // files upload/download
          /ini
          /ini/incremental
          /software
          /cliScript
          /cliScript/incremental
          /...
       /alarms
          /active               // active alarms
          /history              // history alarms
       /license                 // license management
       /performanceMonitoring   // performance monitoring
       /status                  // device status
```

URL

```
/api/v1
```

HTTP Method

```
GET
```

HTTP Response

```
200 OK
```

Example

```
GET /api/v1
HTTP/1.1 200 OK
Content-Type: application/json
{
    "items": [
        {
            "id": "actions",
            "description": "Device actions",
            "url": "/api/v1/actions"
        },
        {
            "id": "alarms",
            "description": "Device alarms",
            "url": "/api/v1/actions"
```

```
        },
        {
            "id": "files",
            "description": "Upload and download of
configuration files",
            "url": "/api/v1/files"
        },
        {
            "id": "license",
            "description": "License management",
            "url": "/api/v1/license"
        },
        {
            "id": "performanceMonitoring",
            "description": "Performance monitoring",
            "url": "/api/v1/performanceMonitoring"
        },
        {
            "id": "status",
            "description": "Device status",
            "url": "/api/v1/status"
        },
    ]
}
```

5 Actions

The `/actions` URL provides the ability to perform maintenance actions on the device.

URL

```
/api/v1/actions
```

HTTP Method

```
GET
```

HTTP Response

```
200 OK
```

Example

```
GET /api/v1/actions

HTTP/1.1 200 OK
Content-Type: application/json
{
    "actions": [
        {
            "id": "reset",
            "description": "Reset device",
            "url": "/api/v1/actions/reset"
        },
        {
            "id": "saveConfiguration",
            "description": "Save device configuration to
NVRAM",
            "url": "/api/v1/actions/saveConfiguration"
        },
    ]
}
```

5.1 Reset Device

The `/actions/reset` URL performs a device reset.

URL

```
/api/v1/actions/reset
```

HTTP Method

POST

Supported JSON Attributes

Attribute	Type	Value	Description
saveConfiguration	Boolean	true	Store current configuration before reset (default).
		false	Do not store current configuration.
gracefulTimeout	Number	0	Perform reset immediately (default).
		1	Wait for all calls to finish, and then perform reset.
		<sec>	Wait for specified time (in seconds) for calls to finish, and then perform reset.

HTTP Responses

- `200 OK`
- `400 Bad request` – provided attributes or values are incorrect.
- `409 Conflict` – reset can't be performed due to current device state (e.g. synchronization with the redundant device is in progress).

Example

```
POST /api/v1/actions/reset
Content-Type: application/json
{
    "saveConfiguration": true,
    "gracefulTimeout": 0
}
HTTP/1.1 200 OK
Content-Type: application/json
{
    "description": "Device will reset now"
}

or

HTTP/1.1 409 Conflict
Content-Type: application/json
{
```

```
    "description": "Device is currently performing HA
synchronization"
}
```

5.2 Save Configuration

The `/actions/saveConfiguration` URL saves the device configuration to the non-volatile memory so that it'll be preserved if the device reboots or is powered down.

URL

```
/api/v1/actions/saveConfiguration
```

HTTP Method

```
POST
```

HTTP Responses

- `200 OK`
- `409 Conflict` – configuration can't be save due to current device state.

Example

```
POST /api/v1/actions/saveConfiguration
HTTP/1.1 200 OK
```

This page is intentionally left blank.

6 Files

The `/files` URL provides access to the various device configuration files.

The PUT method is used to modify the specific configuration file.

The GET method is used to get the specific configuration file (for files which support this method).

URL

```
/api/v1/files
```

HTTP Method

```
GET
```

HTTP Response

```
200 OK
```

Example

```
GET /api/v1/files

HTTP/1.1 200 OK
Content-Type: application/json
{
    "files": [
        {
            "id": "ini",
            "description": "INI configuration file",
            "url": "/api/v1/files/ini"
        },
        {
            "id": "software",
            "description": "Software load",
            "url": "/api/v1/files/software"
        },
        {
            "id": "cliScript",
            "description": "CLI configuration script",
            "url": "/api/v1/files/cliScript"
        },
        ...
    ]
}
```

6.1 INI File

The INI file is the main device configuration file.

6.1.1 Full INI File

The `/files/ini` URL provides the ability to upload or download an ini configuration file. Uploading of an ini file triggers device reset to activate the new configuration. Use `/files/ini/incremental` (see Section 6.1.2) to apply a partial configuration that doesn't require device reset.

URL

```
/api/v1/files/ini
```

HTTP Method

```
GET, PUT
```

Content Type

```
application/octet-stream
```

HTTP Responses

- `200 OK`
- `400 Bad request` - provided ini file is incorrect.
- `409 Conflict` – ini file can't be loaded due to the current device state (e.g. synchronization with the redundant device is in progress).

Example

```
GET /api/v1/files/ini

HTTP/1.1 200 OK
Content-Type: application/octet-stream
<INI file>
```

Example

```
PUT /api/v1/files/ini
Content-Type: application/octet-stream
<INI file>

HTTP/1.1 200 OK
Content-Type: application/json
{
    "description": "Device will reset now to activate new
configuration"
}

or
```

```
HTTP/1.1 409 Conflict
Content-Type: application/json
{
    "description": "Device is currently performing HA
synchronization"
}
```

Note: The uploaded file gets transformed by the device; therefore the file content will differ when you download it.

6.1.2 Incremental INI File

The `/files/ini/incremental` URL provides the ability to upload an incremental (partial) ini file that can be applied to the device without reset.

URL

```
/api/v1/files/ini/incremental
```

HTTP Method

```
PUT
```

Content Type

```
application/octet-stream
```

HTTP Responses

- `200 OK`
- `400 Bad request` - provided ini file is incorrect.
- `409 Conflict` – ini file can't be loaded due to the current device state (e.g. synchronization with the redundant device is in progress).

Example

```
PUT /api/v1/files/ini/incremental
Content-Type: application/octet-stream
<INI file>
HTTP/1.1 200 OK
```

6.2 CLI Script

The CLI configuration script is an alternative method (to the ini file) for detailing the device configuration.

6.2.1 Full CLI Script

The `/files/cliScript` URL provides the ability to upload or download a CLI configuration script. Uploading of a CLI script triggers device reset to activate the new configuration. Use `/files/cliScript/incremental` (see Section 6.2.26.2.2) to apply a partial configuration that doesn't require device reset.

URL

```
/api/v1/files/cliScript
```

HTTP Methods

```
GET, PUT
```

Content Type

```
application/octet-stream
```

HTTP Responses

- `200 OK`
- `400 Bad request` - provided CLI script is incorrect.
- `409 Conflict` – CLI script can't be loaded due to the current device state (e.g. synchronization with the redundant device is in progress).

Example

```
GET /api/v1/files/cliScript

HTTP/1.1 200 OK
Content-Type: application/octet-stream
<CLI script>
```

Example

```
PUT /api/v1/files/cliScript
Content-Type: application/octet-stream
<CLI script>

HTTP/1.1 200 OK
Content-Type: application/json
{
    "description": "Device will reset now to activate new
configuration"
}

or
```

```
HTTP/1.1 409 Conflict
Content-Type: application/json
{
    "description": "Device is currently performing HA
synchronization"
}
```

Note: The uploaded file gets transformed by the device; therefore the file content will differ when you download it.

6.2.2 Incremental CLI Script

The `/files/cliScript/incremental` URL provides the ability to upload an incremental (partial) CLI script to apply to the device without reset. The script may contain both configuration and query commands. Output of the script will be returned in the response.

URL

```
/api/v1/files/cliScript/incremental
```

HTTP Method

```
PUT
```

Content Type

```
application/octet-stream
```

HTTP Responses

- `200 OK`
- `400 Bad request` – provided CLI script is incorrect.
- `409 Conflict` – CLI script can't be loaded due to current device state (e.g. synchronization with the redundant device is in progress).

Example

```
PUT /api/v1/files/cliScript/incremental
Content-Type: application/octet-stream
<show system uptime>
HTTP/1.1 200 OK
Content-Type: application/json
{
    "description": "Incremental CLI script file was loaded",
    "output": "Uptime: 1 days, 0 hours, 36 minutes, 40 seconds"
}
```

6.3 Software Load

The `/files/software` URL provides the ability to modify the device software load. Uploading of the software load triggers a device reset to activate it.

URL

```
/api/v1/files/software
```

HTTP Method

```
PUT
```

Content Type

```
application/octet-stream
```

HTTP Responses

- `200 OK`
- `400 Bad request` – provided software load is incorrect.
- `409 Conflict` – software load can't be applied due to the current device state (e.g. synchronization with the redundant device is in progress).

Example

```
PUT /api/v1/files/software
Content-Type: application/octet-stream
<software load>
HTTP/1.1 200 OK
Content-Type: application/json
{
    "description": "Device will reset now to activate new
software load"
}

or

HTTP/1.1 409 Conflict
Content-Type: application/json
{
    "description": "Device is currently performing HA
synchronization"
}
```

6.3.1 Hitless Software Upgrade

The `/files/software/hitless` URL provides the ability to upgrade the software load on an HA system via the "hitless" procedure (without service interruption).

URL

```
/api/v1/files/software/hitless
```

HTTP Method

```
PUT
```

Content Type

```
application/octet-stream
```

HTTP Responses

- `200 OK`
- `400 Bad request` – provided software load is incorrect.
- `409 Conflict` – software load can't be applied due to the current device state (e.g. synchronization with the redundant device is in progress).

Example

```
PUT /api/v1/files/software/hitless
Content-Type: application/octet-stream
<software load>
HTTP/1.1 200 OK
Content-Type: application/json
{
    "description": "Device will perform switchover to activate
new software load"
}

or

HTTP/1.1 409 Conflict
Content-Type: application/json
{
    "description": "Device is currently performing HA
synchronization"
}
```

6.4 Auxiliary Files

Auxiliary files can be loaded to the device in a similar manner to the software loading URLs that are described in Section 6.3. The following auxiliary files are supported:

- `/files/amd` – answering machine detection
- `/files/castable` – CAS table.
- `/files/codertable` – external coders table.
- `/files/cpt` – call progress tones.
- `/files/dialplan` – dialing plan.
- `/files/prt` – pre-recorded tones.
- `/files/voiceprompts` – voice prompts.

7 Alarms

The `/alarms` URL provides the ability to retrieve the device active and history alarms.

URL

```
/api/v1/alarms
```

HTTP Method

```
GET
```

HTTP Response

```
200 OK
```

Example

```
GET /api/v1/alarms

HTTP/1.1 200 OK
Content-Type: application/json
{
    "alarms": [
        {
            "id": "active",
            "description": "Active alarms",
            "url": "/api/v1/alarms/active"
        },
        {
            "id": "history",
            "description": "History alarms",
            "url": "/api/v1/alarms/history"
        }
    ]
}
```

7.1 Active Alarms

The `/alarms/active` URL provides the ability to retrieve active device alarms.

URL

```
/api/v1/alarms/active
```

HTTP Method

```
GET
```

Supported Parameters

Parameter	Type	Description
?limit=<value>	Number	Limits response to a specified number of alarms. Note that the device may return fewer alarms – e.g. if no more alarms exist or if the user-specified number is too large. Default = 20.
?after=<value>	as returned in previous response	Returns alarms after the alarm specified by the cursor. The cursor value should be taken from "cursor" element in the previous response.
?before=<value>	as returned in previous response	Returns alarms before the alarm specified by the cursor (backwards search). The cursor value should be taken from the "cursor" element in the previous response.

HTTP Responses

- `200 OK`
- `204 No Content` – when no alarms are found

Example

```
GET /api/v1/alarms/active
HTTP/1.1 200 OK
Content-Type: application/json
{
    "alarms": [
        {
            "id": "1",
            "description": "Trunk is down",
            "url": "/api/v1/alarms/active/1"
        },
        {
            "id": "2",
            "description": "Device will explode in 15 min",
            "url": "/api/v1/alarms/active/2"
        }
```

```
    ],
    "cursor": {
        "after": "2",
        "before": "-1"
    }
}
```

The `200 OK` response includes the “cursor” structure that includes “before” and “after” cursors that may be used in consequent requests. Value “-1” indicates than no more alarms before or after exist.

Example

```
GET /api/v1/alarms/active?after=2
HTTP/1.1 200 OK
Content-Type: application/json
{
    "alarms": [
        {
            "id": "3",
            "description": "Intrusion detected",
            "url": "/api/v1/alarms/active/3"
        }
    ],
    "cursor": {
        "after": "-1",
        "before": "3"
    }
}
```

Example

```
GET /api/v1/alarms/active?after=3
HTTP/1.1 204 No Content
```

7.1.1 Specific Active Alarm

Use the following URL to retrieve a specific active alarm.

URL

```
/api/v1/alarms/active/<id>
```

HTTP Method

```
GET
```

HTTP Responses

- `200 OK`
- `404 Not Found` – when alarm is not found

Example

```
GET /api/v1/alarms/active/1
HTTP/1.1 200 OK
Content-Type: application/json
{
    "id": "1",
    "description": "Trunk is down",
    "severity": "Major",
    "source": "Board#1",
    "date": "2010-03-01T23:00:00.000Z",
    "url": "/api/v1/alarms/active/1"
}
```

7.2 History Alarms

The `/alarms/history` URL provides the ability to retrieve device alarms history, including all alarms raised and cleared by the device since the last reboot.

URL

```
/api/v1/alarms/history
```

HTTP Method

```
GET
```

Supported Parameters

Parameter	Type	Description
?limit=<value>	Number	Limits response to a specified number of alarms. Note that the device may return fewer alarms – e.g. if no more alarms exist or if the user-specified number is too large. Default = 20.
?after=<value>	As returned in previous response	Returns alarms after the alarm specified by the cursor. The cursor value should be taken from the “cursor” element in the previous response.
?before=<value>	As returned in previous response.	Returns alarms before the alarm specified by the cursor (backwards search). The cursor value should be taken from the “cursor” element in the previous response.

HTTP Responses

- `200 OK`
- `204 No Content` – when no alarms are found

Example

```
GET /api/v1/alarms/history
HTTP/1.1 200 OK
Content-Type: application/json
{
    "alarms": [
        {
            "id": "1",
            "description": "Trunk is down",
            "url": "/api/v1/alarms/active/1"
        },
        {
            "id": "2",
            "description": "Device will explode in 15 min",
            "url": "/api/v1/alarms/active/2"
```

```
        }
    ],
    "cursor": {
        "after": "2",
        "before": "-1"
    }
}
```

The 200 OK response includes a “cursor” structure that includes “before” and “after” cursors that may be used in consequent requests. The value “-1” indicates than no more alarms before or after exist.

7.2.1 Specific History Alarm

Use the following URL to retrieve a specific history alarm.

URL

```
/api/v1/alarms/history/<id>
```

HTTP Method

```
GET
```

HTTP Responses

- `200 OK`
- `404 Not Found` – when alarm is not found

Example

```
GET /api/v1/alarms/history/1
HTTP/1.1 200 OK
Content-Type: application/json
{
    "id": "1",
    "description": "Trunk is down",
    "severity": "Major",
    "source": "Board#1",
    "date": "2010-03-01T23:00:00.000Z",
    "url": "/api/v1/alarms/history/1"
}
```

8 Device Status

The `/status` URL displays the device status summary.

URL

```
/api/v1/status
```

HTTP Method

```
GET
```

HTTP Response

```
200 OK
```

Example

```
GET /api/v1/status
```

```
HTTP/1.1 200 OK
Content-Type: application/json
{
    "localTimeStamp": "2010-01-17T17:29:15.000Z",
    "ipAddress": "10.4.219.62",
    "subnetMask": "255.255.0.0",
    "defaultGateway": "10.4.0.1",
    "productType": "Mediant SW",
    "versionID": "7.00A.018.005",
    "protocolType": "SIP",
    "operationalState": "UNLOCKED",
    "highAvailability": "Stand Alone"
}
```

This page is intentionally left blank.

9 Performance Monitoring

The `/performanceMonitoring` URL provides access to performance measurements (PMs) collected by the device. The specific PM names are identical to those used in the SNMP interface.

URL

```
/api/v1/performanceMonitoring
```

HTTP Method

```
GET
```

HTTP Response

```
200 OK
```

Example

```
GET /api/v1/performanceMonitoring
HTTP/1.1 200 OK
Content-Type: application/json
{
    "items": [
        "ActiveContextCount",
        "NetUtilKBytes",
        "NetUtilPackets",
        ...
     ]
}
```

9.1 Specific Performance Monitor

Use the following URL to retrieve the specific PM data.

URL

```
/api/v1/performanceMonitoring/<pm>
```

HTTP Method

```
GET
```

Supported Parameters

Parameter	Type	Description
<none>		Returns a list of supported PM indexes and intervals.
?index=<value>	Number	Returns the PM value for the specified index. Value of the index is PM-dependant – e.g. for IP Group PMs it corresponds to the IP Group ID. Refer to the SNMP Alarms Guide to determine the index meaning. Supported indexes can be discovered by issuing a GET request with no parameters.
?interval=<value>	Number	Returns the PM value for the specified interval. • 0 – current (real-time) value (default) ▪ 1 – average value in last 15 minute interval. ▪ 2 – average value in previous 15 minute interval. Supported intervals can be discovered by issuing a GET request with no parameters.

HTTP Responses

- `200 OK`
- `404 Bad Request` – when invalid parameters or values are specified.
- `404 Not Found` – when no PMs are found.

Example

```
GET /api/v1/alarms/performanceMonitoring/netUtilPackets
HTTP/1.1 200 OK
Content-Type: application/json
{
    "indexes": [0, 1],
    "intervalsPerIndex": [0, 1, 2],
    "lowThreshold": 100,
    "highThreshold": 30000
}
```

Example

```
GET /api/v1/alarms/performanceMonitoring/netUtilPackets?index=0
HTTP/1.1 200 OK
Content-Type: application/json
{
    "index": 0,
    "interval": 0,
    "value": -1,
    "min": 0,
    "max": 0,
    "average": 0,
    ...
}
```

Example

```
GET /api/v1/alarms/performanceMonitoring/netUtilPackets?index=100
HTTP/1.1 400 Bad Request
Content-Type: application/json
{
    "description": "Invalid index",
}
```

This page is intentionally left blank.

10 License Management

The `/license` URL provides the ability to view and modify the device license key.

URL

```
/api/v1/license
```

HTTP Method

`GET, PUT`

Content Types

- application/json – not supported for HA configurations (see description of supported attributes below).
- application/octet-stream – supported for all configurations; may include multiple license keys and the device will apply the relevant key based on the corresponding serial number. In an HA configuration, the license may be applied to both the active and redundant devices.

Supported JSON Attributes

Attribute	Type	Value	Description
licenseVersion	Number	1	Indicates the license version. Only version 1 is currently supported.
serialNumber	String		Indicates the license key serial number. If specified number is discovered as different to the device's serial number i.e. a mismatch is found, the update request is rejected. Note that this attribute is not mandatory.
key	String		Indicates the license key in encrypted format.

HTTP Responses

- `200 OK`
- `400 Bad request` - provided license key is incorrect.
- `409 Conflict` – license key can't be loaded due to the current device state (e.g. application/json content type is used for HA device).

Example

```
GET /api/v1/license

HTTP/1.1 200 OK
Content-Type: application/json
{
   "licenseVersion": 1,
   "serialNumber": "277522263687112",
   "key": "jCx6r5tovCIKaBBbhPtT53Yj",
   "keyDescription": "Key features: Board Type: Mediant 800
```

```
Security: IPSEC MediaEncryption StrongEncryption
EncryptControlProtocol
Coders: G723 G729 G728 NETCODER GSM-FR GSM-EFR AMR EVRC-QCELP
G727 ILBC EVRC-B “
  }
```

Example

```
PUT /api/v1/license
Content-Type: application/json
{
   "licenseVersion": 1,
   "serialNumber": "277522263687112",
   "key": "jCx6r5tovCIKaBBbhPtT53Yj"
}
HTTP/1.1 200 OK
Content-Type: application/json
{
    "description": "Device will reset now to activate new
license"
}

or

HTTP/1.1 409 Conflict
Content-Type: application/json
{
    "description": "License key can’t be applied to device in
HA configuration. Use license file instead."
}
```

Example

```
PUT /api/v1/license
Content-Type: application/octet-stream
<license file>
HTTP/1.1 200 OK
Content-Type: application/json
{
    "description": "Device will reset now to activate new
software load"
}
```

11 Supported HTTP Responses

The following HTTP responses are used by the REST API:

- `200 OK` – indicates successful request completion.
- `201 Created` – indicates the creation of a new resource.
- `204 No Content` – indicates that no items are found in response to a discovery request.
- `400 Bad Request` – indicates a request failure due to an invalid input.
- `401 Unauthorized` – indicates a request failure due to incorrect authentication credentials.
- `403 Forbidden` – indicates a request failure due to an authorization failure (i.e. URL exists; however the user is not authorized to access it).
- `404 Not Found` – indicates an invalid URL.
- `405 Method Not Allowed` – indicates that the HTTP method is not supported on the specific URL/resource.
- `406 Not Acceptable` – indicates that the client included “Accept:” header in a request that doesn’t include the format used by the server (for most URLs it’s “application/JSON”).
- `409 Conflict` – indicates a failure due to “intermittent” reason (e.g. synchronization with the redundant device is in progress).
- `500 Internal Server Error` – indicates an internal failure.

Mediant REST API

www.audiocodes.com