



API Reference

Conductor Live



Version 3.22.0 and later

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Conductor Live: API Reference

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

About This Manual	1
Working with the API	2
The API Protocol	2
Entities, Attributes, Elements, Properties, Parameters	2
Requests	3
Request URLs	3
Header Content - Standard Elements	4
Header Content for User Authentication	5
Body Content	5
Encoding String Parameters in the URL Request	6
Versioning	6
Case Sensitivity of Names and Values	6
Boolean Values in Attributes	6
Arrays	6
Null Values	7
Using the API with User Authentication Enabled	7
Hashing the API Key	8
AuthCurl Scripts	8
Authentication Error Messages	9
Node Changes with SSL Enabled	10
"Clean" Requests	10
Responses	10
Content of Responses	10
Success Response	11
Error Response	11
IDs of Entities	11
Obtaining an ID	11
Multiple Identities	12
Uniqueness of IDs	12
Commands	13
Profiles	13
Channels	14
Channel Schedules	14
Bulk Tasks	15

MPTS	16
Members of an MPTS	17
Nodes	18
Router	18
Router Inputs	19
Router Outputs	20
Redundancy Groups	21
Members of a Redundancy Group	21
Conductor Redundancy Groups	22
Members of a Conductor Redundancy Group	23
Pass Through to AWS Elemental Live	24
Passing Through to AWS Elemental Live	26
Passthrough of Live Event POST Commands	26
HTTP Request and Response	26
Example	29
Passthrough of Live Event GET Commands	30
HTTP Request and Response	30
Example	31
Passthrough of Live System Status	31
HTTP Request and Response	31
Example	32
Validating Your Generated XML	33
Working with the Cluster	34
Working with Profiles	34
Recommended Method for Working with Profiles	34
POST: Create a Profile	38
Modify a Profile	40
GET List: Get a List of Profiles	40
GET: Get the Attributes of a Profile	42
DELETE: Delete a Profile	43
Working with Channels	43
POST: Create a Channel	44
PUT: Modify the Attributes of a Channel	48
GET List: Get List of Channels	49
GET: Get the Attributes of a Channel	52
DELETE: Delete a Channel	54

Channel Scheduling	54
CRON Syntax Summary	55
POST: Create a One-Time Schedule	58
POST: Create a Repeating Schedule	60
POST: Activate a Schedule	63
DELETE: Deactivate a Schedule	63
PUT: Update a Schedule	64
GET List: Get List of All Channel Schedules	65
GET: Get the Attributes of a Schedule	68
GET: Get Schedule Events	69
DELETE: Delete a Schedule	71
Performing Bulk Tasks on a Channel	72
POST Start: Start One or More Channels	72
POST Stop: Stop One or More Channels	74
Monitoring Bulk Tasks: GET List of Task Reports	75
Monitoring Bulk Tasks: GET One Task Report	79
Controlling Ad Avail on a Channel	82
POST Ad Avail State: Start or Stop Ad Avail on a Channel	82
Working with MPTS	83
POST: Create an MPTS	83
PUT: Modify the Attributes of an MPTS	95
GET List: Get a List of MPTS Outputs	95
GET: Get the Attributes of an MPTS Output	98
DELETE: Delete an MPTS Output	99
GET Status List: Get the Status of a List of MPTS Outputs	100
GET Status: Get the Status of an MPTS Output	101
GET Bitrate: Get the Bitrate of an MPTS Output	101
POST Start: Start an MPTS	105
DELETE Stop: Stop an MPTS	106
PUT: Swap Allocation	107
Working with Members of an MPTS	108
POST: Add an SPTS to an MPTS	109
PUT: Modify an SPTS Program	114
PUT: MPTS Channel Swap	115
GET List: Get All SPTS of an MPTS	122
GET: Get an SPTS Program	129

DELETE: Delete an SPTS Program	130
Monitoring Conductor Live	132
Managing Channels	132
Channel Status Elements	132
POST Channel Revert: Resetting the Channel	133
Querying Alerts and Messages	134
GET Alerts: Get a List of Alerts	134
GET Messages: Get a List of Messages	144
GET System Information: Get a List of System Details	150
Configuring the Cluster	155
Setting up Nodes	155
POST: Add a Node to the Cluster	155
GET List: Get a List of Nodes in the Cluster	157
GET: Get the Attributes of a Node	162
DELETE: Remove a Node from the Cluster	163
Setting Up Routers	164
POST: Create a Router	164
PUT: Modify a Router	167
GET List: Get a List of Routers	168
GET: Get Router Attributes	171
DELETE: Delete a Router	172
Setting Up Router Inputs	173
POST: Create a Router Input	173
PUT: Modify a Router Input	176
GET List: Get a List of Router Inputs	177
GET: Get Attributes of a Router Input	179
DELETE: Delete a Router Input	181
Setting Up Router Outputs	181
POST: Create a Router Output	182
PUT: Modify a Router Output	185
GET List: Get Router Output List	186
GET: Get Attributes of a Router Output	189
DELETE: Delete a Router Output	190
Setting Up Redundancy Groups	190
POST: Create a Redundancy Group	190
PUT: Modify a Redundancy Group	192

GET List: Get a List of Redundancy Groups	193
GET: Get Attributes of a Redundancy Group	195
DELETE: Delete a Redundancy Group	196
Setting Up Members of a Redundancy Group	196
POST: Add a Node to a Redundancy Group	197
PUT: Change Role of a Member of a Redundancy Group	200
GET List: Get a List of Redundancy Group Members	201
GET: Get the Attributes of a Redundancy Group Member	204
DELETE: Remove a Node from a Redundancy Group	205
POST Initiate Failover	205
Setting Up a Conductor Redundancy Group	206
POST: Create a Conductor Redundancy Group	206
PUT: Modify a Conductor Redundancy Group	209
GET: Get the Attributes of the Conductor Redundancy Group	210
DELETE: Delete a Conductor Redundancy Group	212
POST Enable: Enable Conductor Redundancy Group	212
DELETE Disable: Disable Conductor Redundancy Group	214
Setting Up Members of a Conductor Redundancy Group	216
POST: Add a Node to a Conductor Redundancy Groups	216
PUT: Modify a Member of a Conductor Redundancy Group	218
GET List: Get a List of Conductor Redundancy Group Members	218
GET: Get the Attributes of a Conductor Redundancy Group Member	220
DELETE: Remove a Node from the Conductor Redundancy Group	221
Backing Up the Conductor Database	222
PUT: Modify Database Backup Settings	222
POST: Backup Database Now	223
Document History	225

About This Manual

This guide is intended for operators who will set up and create nodes, redundancy groups, channels, and profiles using the Conductor Live REST API and for operators who run and manage activity on all the AWS Elemental Live and AWS Elemental Statmux nodes in a cluster using the REST API.

It is assumed that you are familiar with:

- Working with a REST API and have selected a REST client to use.
- The Conductor Live constructs. If you are not, see the [AWS Elemental Conductor Live User Guide](#).

Related Documentation

For additional information, see the following:

- Conductor Live User Guide
- AWS Elemental Live User and API Guide
- AWS Elemental Statmux User Guide

Working with the API

Topics

- [The API Protocol](#)
- [Requests](#)
- [Using the API with User Authentication Enabled](#)
- [Node Changes with SSL Enabled](#)
- [“Clean” Requests](#)
- [Responses](#)
- [IDs of Entities](#)

The API Protocol

The Conductor Live API can be accessed using HTTP or HTTPS. The API follows the REST architectural framework. In accordance with REST guidelines, the API exposes four types of operations based on the requesting HTTP(S) method:

- POST
- GET
- PUT
- DELETE

Entities, Attributes, Elements, Properties, Parameters

The entities that the Conductor Live API works with are:

- Redundancy groups
- Channels
- Nodes
- Profiles
- Schedules
- MPTS outputs

- MPTS members (SPTS programs)
- Routers

These entities have attributes in Conductor Live . In the API, these attributes are passed in the XML body of the request or response. They are passed as either:

- XML elements (if they are read-write), or
- Properties of an XML element (if they are read-only).

API requests may have parameters, which are presented in angle brackets. For example, <ID of redundancy group> is a parameter in this POST request:

```
/redundancy_groups/<ID of redundancy group>/members
```

Requests

Requests consist of a URL, a header, and a body.

Topics

- [Request URLs](#)
- [Header Content - Standard Elements](#)
- [Header Content for User Authentication](#)
- [Body Content](#)
- [Encoding String Parameters in the URL Request](#)
- [Versioning](#)
- [Case Sensitivity of Names and Values](#)
- [Boolean Values in Attributes](#)
- [Arrays](#)
- [Null Values](#)

Request URLs

The request consists of the operation, the IP address of the Conductor Live node, and resources in a parent/child structure. For example:

```
POST http://198.51.100.0/redundancy_groups/3/members
```

In this example, the URL refers to members of the third redundancy group. That is, “redundancy groups” is the parent of each redundancy group, “3” is the ID of a particular redundancy group. The redundancy group “3” is the parent of “members”, which is the group of all members of the redundancy group 3. This POST command would contain an xml body as a child of “members” that would represent a particular member of this group.

Specifying Pagination of the Response

For responses that include large amounts of data, use pagination in the request. When you do so, your data is returned grouped into “pages” with the specified number of elements per page.

To use pagination, append your request with “?page=x&per_page=y”, where x is the number of the page you want to see and y is the number of items shown per page. For example:

```
GET http://198.51.100.0/channels?page=2&per_page=15
```

will return page 2 of the list of channels presented at 15 channels per page. In this example, 198.51.100.0 is the IP address of the Conductor Live system that is managing the channels.

If you include just “page”, the system will use the default of 20 items per page. If you include just “per_page”, the system will use the default of 1 for page. If you do not include either, the system will return all available items at once.

Note

Data returned is not necessarily ordered chronologically. Therefore, do not use pagination as a filter to locate recent data.

Header Content - Standard Elements

For All Requests

- Accept: Set to application/xml

For PUT and POST Requests

- Accept: Set to application/xml
- Content-Type: Set to application/xml

When POSTing xml for a profile originally created with an earlier version of Conductor Live , set content-type to application/vnd.elemental+xml;version=n.n.n, where n.n.n is the number of the Conductor Live version used to create the profile. For example, if the profile was created with Conductor Live version 3.2.1, use:

```
content-type: application/vnd.elemental+xml;version=3.2.1
```

For more information on using profiles created with earlier software versions, see [the section called “Versioning”](#).

Header Content for User Authentication

If your cluster deployment is configured for user authentication (users must log into Conductor Live), then the header must also include:

- X-Auth-User header.
- X-Auth-Expires header (optional).
- X-Auth-Key header includes the API key of the individual user.

For more information, see [the section called “Using the API with User Authentication Enabled”](#).

Body Content

The body, if required, consists of XML content. The body is:

- Required for most POST requests.
- Required for all PUT requests.
- Not required for a GET or DELETE.

Encoding String Parameters in the URL Request

Note

All string parameters in the URL request must be UTF-8 encoded. String parameters containing non-ASCII characters must be URL-encoded.

Versioning

Compatibility between XML and Conductor Live Client

Starting with version 3.0.3, Conductor Live can accept XML that was generated using a software release that is up to 2 versions older.

When submitting a request, specify the version of the XML in the header of the request; see above. When the version is included, the contents of the XML is checked:

- If there are elements that are required in the current version of the client and they are not in the XML, the client adds them and assigns the default value.
- If there are elements that are not understood by the current version of Conductor Live, AWS Elemental Live, or AWS Elemental Statmux, the request is rejected.

Compatibility between Conductor Live and AWS Elemental Live/Statmux

Each Conductor Live version is compatible with a specific range of AWS Elemental Live or AWS Elemental Statmux versions. For example, an AWS Conductor Live version 3.6.x is compatible with AWS Elemental Live API version 2.13.x for any x.

Case Sensitivity of Names and Values

The names and the values of all Conductor Live attributes are case sensitive.

Boolean Values in Attributes

Boolean values in attributes must be entered as "true" or "false". 0 and 1 are not acceptable values.

Arrays

In a POST or PUT, an array must include the property **type="array"**. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<hosts type="array">
  <host>10.4.136.15</host>
  <host>10.4.136.[90-92]</host>
</hosts>
```

Null Values

A null value is not the same as an empty string. To set a null value for an attribute, enter it as follows:

```
<error_clear_time nil="true"/>
```

Using the API with User Authentication Enabled

Your cluster deployment is configured for local or PAM user authentication (users must provide valid credentials to access Conductor Live). Check with the person who performed the initial configuration of the cluster, or see the [AWS Elemental Conductor Live Configuration Guide](#).

If authentication is enabled, then the header of each request must also include the following:

Header	Description
X-Auth-User	The username of the user who is using the API. Note that the user's password is not included in the header.
X-Auth-Expires	The date and time at which the individual REST request expires. Enter the date in Unix time (POSIX or Epoch time). The recommended value is 30 seconds in the future, but, if the client clock and Conductor node clock are not completely in sync, you may want to make adjustments to accommodate the difference.
X-Auth-Key	An MD5 hash of the API key for the user who is using the API.

Header	Description
	An administrator generates this key as follows: • Log on via the web interface and go to Settings > User Profile. • Click the Reset API Key (key icon) for the applicable users. • Provide the individual user with a key, for example, via email. For information on hashing the key, see the next section.

Topics

- [Hashing the API Key](#)
- [AuthCurl Scripts](#)
- [Authentication Error Messages](#)

Hashing the API Key

Construct the X-Auth-Key header as follows:

```
md5(api_key + md5(url + X-Auth-User + api_key + X-Auth-Expires))
```

- The + operator indicates string concatenation without any delimiters.
- Enter each parameter in this expression as a string.
- The url parameter is the path portion of the request URL minus any query parameters and without any API version prefix. It must not have a trailing slash.

The hash is valid for a single access: it is not persisted.

AuthCurl Scripts

Two helper scripts are available to help construct these headers:

- `auth_curl.rb`
- `auth_curl.pl`

These scripts are located in the following directory.

```
/opt/elemental_se/web/public/authentication_scripts
```

Authentication Error Messages

The following errors describe why authentication requests can fail.

Error Message	Reason
X-Auth-Login is required.	The request headers are missing "X-Auth-Login" or the values are malformed.
X-Auth-Key is required.	The request headers are missing "X-Auth-Key" or the values are malformed.
X-Auth-Expires is required.	The request headers are missing "X-Auth-Expires" or the values are malformed.
X-Auth-Login is invalid.	Invalid user or the user exists but the role is invalid.
X-Auth-Key is invalid.	The key is not valid. To troubleshoot: • Double-check the token generation. • Ensure UTC Time is used. • Verify the endpoint URL. • Verify the api-key.
X-Auth-Expires is invalid.	The "expires" value did not pass validation. To troubleshoot: • Ensure UTC Time is used.

Error Message	Reason
	• Confirm that the server and client time is in sync with an NTP server. • Increase the delta value (default is 30ms) on the client when issuing the request.

Node Changes with SSL Enabled

When SSL (HTTPS) is enabled, the `--https` command must be used whenever a node is reconfigured. Without the command, SSL is disabled.

Throughout this guide, use `https://` instead of `http://` if you have SSL enabled.

“Clean” Requests

A quick way to prepare the body for a new POST is to do a GET on an existing entity, and include the `clean` parameter in the request. With this parameter set to `true`, the GET response omits the `<id>` elements and other elements such as `<status>`, `<service_name>`, and `<service_provider_name>` that do not apply to a POST.

1. Append `?clean=true` to the GET command. For example:

```
GET http://101.4.136.95/profiles/2.xml?clean=true
```

2. Modify the response as desired and paste the revision in the body of the desired POST request.

Responses

Content of Responses

Responses consist of a header and a body.

The header always contains two elements:

-
- Content-Type: Set to `application/xml`.

- **Accept:** Set to application/xml. For PUSH and POST requests only
- The body consists of XML content. The body contains:
 - Unsuccessful request: a description of the error.
 - Successful POST or PUT request: the ID of the entity that was created or changed and a summary.
 - Successful GET request: the requested content.
 - Successful DELETE request: present but empty.

Success Response

If a request is valid, Conductor Live returns the appropriate response:

- For a POST, PUT or DELETE: A 200 OK response. The body may be empty or may contain XML content.
- For a GET: A 200 OK response with XML content in the body.

Error Response

- If a request is not valid (for example, the request or the body are badly formatted), Conductor Live returns the appropriate HTTP response, typically an error in the 4xx or 5xx range.
- If the URL of the request is invalid (for example, the IP address is wrong), then Conductor Live returns a 404 response.
- If the request is valid (Conductor Live understands it), but the request cannot be fulfilled for some reason, then Conductor Live returns a 422 error.

IDs of Entities

When an entity is created, it is automatically assigned an ID that is stored in the `<id></id>` element.

These unique IDs are typically shown on the Conductor Live web interface under the “ID” column.

Obtaining an ID

The ID is shown in the POST response and can be obtained using GET List. To obtain an ID:

- Obtain a list of IDs for an entity using a GET request.

- Parse the response for the desired ID by looking for the ID that corresponds to a piece of data that you specified, such as the entity name.

The ID must be passed in any PUT, GET, and DELETE. In general, you cannot identify an entity using the name element.

Multiple Identities

Conductor systems and worker nodes within a cluster are assigned a node ID. Redundancy group membership assigns a redundancy member ID. It is important that you not conflate the node ID with the redundancy member ID; they are different IDs.

Any element that is a grouping of attributes is usually assigned a unique ID. The presence of this ID does not mean you can query this grouping by passing in this ID. You can only query the entities listed in [the section called “Entities, Attributes, Elements, Properties, Parameters”](#).

Uniqueness of IDs

Each type of entity has its own numbering scheme. For example, redundancy groups are numbered from 1 and channels are numbered separately, starting from 1.

Numbering increments indefinitely. If an entity is deleted, its number is *not* recycled.

Conductor Live Commands

Topics

- [Profiles](#)
- [Channels](#)
- [Channel Schedules](#)
- [Bulk Tasks](#)
- [MPTS](#)
- [Members of an MPTS](#)
- [Nodes](#)
- [Router](#)
- [Router Inputs](#)
- [Router Outputs](#)
- [Redundancy Groups](#)
- [Members of a Redundancy Group](#)
- [Conductor Redundancy Groups](#)
- [Members of a Conductor Redundancy Group](#)
- [Pass Through to AWS Elemental Live](#)

Profiles

Nickname	Action	Signature	Description
POST Profile	POST	/profiles	Create a profile.
GET Profile List	GET	/profiles	Get the list of profiles.
GET Profile	GET	/profiles/<ID of profile>	Get the attributes of the specified profile.
DELETE Profile	DELETE	/profiles/<ID of profile>	Delete the specified profile.

Channels

Nickname	Action	Signature	Description
POST Channel	POST	/channels	Create a new channel.
PUT Channel	PUT	/channels/<ID of channel>	Modify the attributes of the specified channel.
GET Channel List	GET	/channels	Get the list of channels.
GET Channel	GET	/channels/<ID of channel>	Get the attributes of the specified channel.
DELETE Channel	DELETE	/channels/<ID of channel>	Delete the specified channel.

Channel Schedules

Nickname	Action	Signature	Description
POST	POST	/channels/<channel ID>/schedules	Create a repeating or one-time schedule.
POST Activate Schedule	POST	/channels/<channel ID>/schedules/<schedule ID>/active	Activate a schedule.
DELETE Deactivate Schedule	DELETE	/channels/<channel ID>/schedules/<schedule ID>/active	Deactivate a schedule.
PUT Update Schedule	PUT	/channels/<channel ID>/schedules/<schedule ID>	Modify the attributes of the specified schedule.

Nickname	Action	Signature	Description
GET Schedule List	GET	/channels/<channel ID>/schedules	Get the list of all schedules for a channel.
GET Schedule	GET	/channels/<channel ID>/schedules/<schedule ID>	Get the attributes of the specified schedule.
GET Schedule Events (All)	GET	/events	Get all schedule events for the cluster.
GET Schedule Events (One Schedule)	GET	/channels/<channel ID>/schedules/<schedule ID>/events	Get all schedule events generated from one schedule.
DELETE Delete Schedule	DELETE	/channels/<channel ID>/schedules/<schedule ID>	Delete the specified schedule.

Bulk Tasks

Nickname	Action	Signature	Description
POST Start Channel	POST	/channels/start	Start one or more channels.
POST Stop Channel	POST	/channels/stop	Stop one or more channels.
GET Task Report List	GET	/task_reports/<ID of report>	Get the list of task reports.
GET Task Report	GET	/task_reports/<ID of report>	Get the specified task report.

MPTS

Nickname	Action	Signature	Description
POST MPTS	POST	/mpts	Create an MPTS output and optionally specify its Single-Protocol Transport Service (SPTS) programs.
PUT MPTS	PUT	/mpts/<ID of MPTS>	Change the attributes and/or SPTS programs of the specified MPTS output.
GET MPTS List	GET	/mpts	Get the list of MPTS outputs.
GET MPTS	GET	/mpts/<ID of MPTS>	Get the attributes and SPTS programs of one MPTS output.
DELETE MPTS	DELETE	/mpts/<ID of MPTS>	Delete the specified MPTS output.
GET MPTS Status List	GET	/mpts/statuses	Get the status of all MPTS outputs.
GET MPTS Status	GET	/mpts/<ID of MPTS>/status	Get the status of the specified MPTS output.
GET MPTS Bitrate	GET	/mpts/<ID of MPTS>/stats	Get bitrate information for the specified MPTS output.

Nickname	Action	Signature	Description
POST Start MPTS	POST	/mpts/<ID of mpts>/mux	Start an MPTS output.
DELETE Stop MPTS	GET	/mpts/<ID of mpts>/mux	Stop an MPTS output.
PUT Swap Allocation	PUT	/mpts/swap_allocation_priority	Swap the allocation values (allocation_message_priority element) in two related MPTS outputs.

Members of an MPTS

Nickname	Action	Signature	Description
POST MPTS Member	POST	/mpts/mpts_id/mpts_members	Add an SPTS to the specified MPTS output.
PUT MPTS Member	PUT	/mpts/mpts_id/mpts_members/:<mpts_member_id>	Change the attributes of the specified SPTS program in the specified MPTS output.
GET MPTS Member List	GET	/mpts/mpts_id/mpts_members	Get the list of all the SPTS programs in the specified MPTS output.
GET MPTS Member	GET	/mpts/mpts_id/mpts_members/:<mpts_member_id>	Get the specified SPTS program from

Nickname	Action	Signature	Description
			the specified MPTS output.
DELETE MPTS Member	DELETE	/mpts/mpts_id/ mpts_members/ :<mpts_member_id>	Delete the specified SPTS program from the specified MPTS output.

Nodes

Nickname	Action	Signature	Description
POST Node	POST	/nodes	Add a node to the cluster.
GET Node List	GET	/nodes	Get the list of the nodes in the cluster.
GET Node	GET	/nodes/<ID of node>	Get the attributes on the specified node.
GET Node Status	GET	/nodes/<ID of node>/system_status	Get the status of the specified node.
DELETE Node	DELETE	/nodes/<ID of node>	Remove the specified node from the cluster.

Router

Nickname	Action	Signature	Description
POST Router	POST	/routers	Create a new router.

Nickname	Action	Signature	Description
PUT Router	PUT	/routers/<ID of router>	Modify the attributes of the specified router.
GET Router List	GET	/routers	Get the list of routers.
GET Router	GET	/routers/<ID of router>	Get the attributes of the specified router.
DELETE Router	DELETE	/routers/<ID of router>	Delete the specified router.

Router Inputs

Nickname	Action	Signature	Description
POST Router Input	POST	/routers/<ID of router>/inputs	Create a new input for the specified router.
PUT Router Input	PUT	/routers/<ID of router>/inputs/<ID of input>	Modify the attributes of the specified input on the specified router.
GET Router Input List	GET	/routers/<ID of router>/inputs	Get the list of inputs for the specified router.
GET Router Input	GET	/routers/<ID of router>/inputs/<ID of input>	Get the attributes of the specified input on the specified router.

Nickname	Action	Signature	Description
DELETE Router Input	DELETE	/routers/<ID of router>/inputs/<ID of input>	Delete the specified input on the specified router

Router Outputs

Nickname	Action	Signature	Description
POST Router Output	POST	/routers/<ID of router>/outputs	Create a new output for the specified router.
PUT Router Output	PUT	/routers/<ID of router>/outputs/<ID of output>	Modify the attributes of the specified output on the specified router.
GET Router Output List	GET	/routers/<ID of router>/outputs	Get the list of outputs for the specified router.
GET Router Output	GET	/routers/<ID of router>/outputs/<ID of output>	Get the attributes of the specified output on the specified router.
DELETE Router Output	DELETE	/routers/<ID of router>/outputs/<ID of output>	Delete the specified output on the specified router

Redundancy Groups

Nickname	Action	Signature	Description
POST Group	POST	/redundancy_groups	Create a new redundancy group.
PUT Group	PUT	/redundancy_groups /<ID of redundancy group>	Modify the specified redundancy group.
GET Group List	GET	/redundancy_groups	Get the list of redundancy groups.
GET Group	GET	/redundancy_groups /<ID of redundancy group>	Get the attributes of the specified redundancy group.
DELETE Group	DELETE	/redundancy_groups /<ID of redundancy group>	Delete the redundancy group that has the specified ID.

Members of a Redundancy Group

Nickname	Action	Signature	Description
POST Member	POST	/redundancy_groups /<ID of redundancy group>/members	Add a new node to the specified redundancy group.
PUT Member	PUT	/redundancy_groups /<ID of redundancy group>/members/ <ID of member node>	Modify the attributes of the specified node in the specified redundancy group.

Nickname	Action	Signature	Description
GET Member List	GET	/redundancy_groups /<ID of redundancy group>/members	Get the list of the nodes in the specified redundancy group.
GET Member	GET	/redundancy_groups /<ID of redundancy group>/members/ <ID of member node>	Get the attributes of the specified node in the specified redundancy group.
DELETE Member	DELETE	/redundancy_groups /<ID of redundancy group>/members/ <ID of member node>	Delete the node with the specified ID from the specified redundancy group.
POST Initiate Failover	POST	/nodes/<ID of node>/redundancy	Test redundancy set up by initiating failover of the specified node.

Conductor Redundancy Groups

Nickname	Action	Signature	Description
POST Group	POST	/conductor_redundancy_groups	Create a new Conductor redundancy group.
PUT Group	PUT	/conductor_redundancy_groups/<ID of group>	Modify the specified Conductor redundancy group.

Nickname	Action	Signature	Description
GET Group List	GET	/conductor_redundancy_groups	Get the list of Conductor redundancy groups.
GET Group	GET	/conductor_redundancy_groups/<ID of group>	Get the attributes of the specified Conductor redundancy group.
DELETE Group	DELETE	/conductor_redundancy_groups/<ID of group>	Delete the Conductor redundancy group that has the specified ID.
POST Enable Group	POST	conductor_redundancy_groups/<ID of group>/enable	Enable redundancy on the two Conductor nodes in the Conductor redundancy group.
DELETE Disable Group	DELETE	conductor_redundancy_groups/<ID of group>/disable	Disable redundancy on the Conductor redundancy group.

Members of a Conductor Redundancy Group

Nickname	Action	Signature	Description
POST Member	POST	/conductor_redundancy_groups/<ID of group>/members	Add a new node to the specified Conductor redundancy group.

Nickname	Action	Signature	Description
GET Member List	GET	/conductor_redundancy_groups/<ID of group>/members	Get the list of the nodes in the specified Conductor redundancy group.
GET Member	GET	/conductor_redundancy_groups/<ID of group>/members/<ID of member node>	Get the attributes of the specified node in the specified Conductor redundancy group.
DELETE Member	DELETE	/conductor_redundancy_groups/<ID of group>/members/<ID of member node>	Delete the node with the specified ID from the specified Conductor redundancy group.
GET Alerts	GET	/alerts	Get a list of the alerts that have occurred.
GET Messages	GET	/messages	Get a list of the messages that have occurred.

Pass Through to AWS Elemental Live

Nickname	Action	Signature	Description
POST Live action	POST	http://<Conductor IP address>/channels/<ID of channel>/live_events/<action>	Pass an AWS Elemental Live API event command to

Nickname	Action	Signature	Description
			the Live node via the Conductor Live API.
GET System Status	GET	http://<Conductor IP address>/nodes/<ID of node>/system_status	Get status information on an AWS Elemental Live node in the cluster.
GET Inputs	GET	http://<Conductor IP address> /channels /<ID of channel>/live_events/inputs	Get the ID of an event input.

Passing Through to AWS Elemental Live

You can use the Conductor Live API to submit some AWS Elemental Live API commands to an AWS Elemental Live node being controlled by this Conductor node.

Topics

- [Passthrough of Live Event POST Commands](#)
- [Passthrough of Live Event GET Commands](#)
- [Passthrough of Live System Status](#)

Passthrough of Live Event POST Commands

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/channels/<ID of channel>/live_events/<action>
```

where:

- ID of channel is the ID of a channel known to the Conductor Live API.
- /live_events/<action> is the command from the AWS Elemental Live API. See the table below for a complete list of commands.

When the command is submitted, the Conductor Live API determines the Live event that corresponds to the Conductor Live channel and then submits the appropriately formed command to the Live API.

Action	Signature in AWS Elemental Live	Signature in Conductor Live
POST	/live_events/<id>/activate_input	/channels/<ID of channel>/live_events/activate_input

Action	Signature in AWS Elemental Live	Signature in Conductor Live
POST	/live_events/<id>/adjust_audio_gain	/channels/<ID of channel>/live_events/adjust_audio_gain
POST	/live_events/<id>/avail_image	/channels/<ID of channel>/live_events/avail_image
POST	/live_events/<id>/blackout_image	/channels/<ID of channel>/live_events/blackout_image
POST	/live_events/<id>/bulk_metadata	/channels/<ID of channel>/live_events/bulk_metadata
POST	/live_events/<id>/cue_point	/channels/<ID of channel>/live_events/cue_point
POST	/live_events/<id>/motion_image_inserter	/channels/<ID of channel>/live_events/motion_image_inserter
POST	/live_events/<id>/mute_audio	/channels/<ID of channel>/live_events/mute_audio
POST	/live_events/<id>/unmute_audio	/channels/<ID of channel>/live_events/unmute_audio
POST	/live_events/<id>/pause_output	/channels/<ID of channel>/live_events/pause_output
POST	/live_events/<id>/unpause_output	/channels/<ID of channel>/live_events/unpause_output
POST	/live_events/<id>/pause_output_group	/channels/<ID of channel>/live_events/pause_output_group

Action	Signature in AWS Elemental Live	Signature in Conductor Live
POST	/live_events/<id>/unpause_output_group	/channels/<ID of channel>/live_events/unpause_output_group
POST	/live_events/<id>/private_metadata	/channels/<ID of channel>/live_events/private_metadata
POST	/live_events/<id>/reset_video_buffer_stats	/channels/<ID of channel>/live_events/reset_video_buffer_stats
POST	/live_events/<id>/rollover_output	/channels/<ID of channel>/live_events/rollover_output
POST	/live_events/<id>/start_output	/channels/<ID of channel>/live_events/start_output
POST	/live_events/<id>/start_output_group	/channels/<ID of channel>/live_events/start_output_group
POST	/live_events/<id>/stop_output	/channels/<ID of channel>/live_events/stop_output
POST	/live_events/<id>/stop_output_group	/channels/<ID of channel>/live_events/stop_output_group
POST	/live_events/<id>/time_signal	/channels/<ID of channel>/live_events/time_signal
POST	/live_events/<id>/timed_metadata	/channels/<ID of channel>/live_events/timed_metadata
POST	/live_events/<id>/image_inserter	/channels/<ID of channel>/live_events/image_inserter

Action	Signature in AWS Elemental Live	Signature in Conductor Live
POST	/live_events/<id>/image_inserter/input	/channels/<ID of channel>/live_events/image_inserter/input
POST	/live_events/<id>/prepare_input	/channels/<ID of channel>/live_events/prepare_input

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

Include a request body only if the original AWS Elemental Live command includes a body. Format the body in exactly the same way.

Response

The response repeats back the response received from the Live API, exactly as received from that API.

Example

AWS Elemental Live REST call:

```
POST http://<Live IP address>/live_events/<ID of event>/mute_audio
```

Conductor Live passthrough of this call:

```
POST http://<Conductor IP address>/channels/3/live_events/mute_audio
```

Passthrough of Live Event GET Commands

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address> /channels/<ID of channel>/live_events/<action>
```

where:

- ID of channel is the ID of a channel known to the Conductor Live API.
- /live_events/<action> is the command from the AWS Elemental Live API. See below for a list.

When the command is submitted, the Conductor Live API determines the Live event that corresponds to the Conductor Live channel, then submits the appropriately formed command to the Live API.

Action	Signature in AWS Elemental Live	Signature in Conductor Live
GET	/live_events/<id>/inputs	/channels/<ID of channel>/live_events/inputs

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

Include a body only if the original AWS Elemental Live command includes a body. Format the body in exactly the same way.

Response

The response repeats back the response received from the Live API, exactly as received from that API.

Example

AWS Elemental Live REST call:

```
GET http://live_events/13/inputs
```

Conductor Live passthrough of this call:

```
GET http://192.0.2.16/channels/13/live_events/inputs
```

Passthrough of Live System Status

You can pass through a request for the status of an AWS Elemental Live system within the Conductor cluster as well. This passthrough command is structured slightly differently from a `live_event` command.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/nodes/<ID of node>/system_status
```

where `<ID of node>` is the ID of the node as assigned by Conductor Live.

When the command is submitted, the Conductor Live API queries the appropriate AWS Elemental Live node and returns the system status as reported by that node.

Call Header

- Accept: Set to `application/xml`
- Content-Type: Set to `application/xml`

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

There is no body in the system_status command.

Response

The response repeats back the response received from the Live API, exactly as received from that API.

Example

```
GET http://198.51.100.0/nodes/13/system_status
```

Validating Your Generated XML

If you have written a script to automatically generate your profiles in xml, validate your output as follows.

To validate your generated XML

1. Generate a profile from your code and save it in your current directory.
2. Copy this .xsd file into your current directory: /opt/elemental_se/web/public/schema/Live247Profile.xsd.
3. Run the following command against your generated profile:

```
xmllint --sax -noout -valid --schema Live247Profile.xsd <your xml filename>
```

The system returns a response indicating whether the profile does or does not validate.

Working with the Cluster

The entities in this chapter are those that you work with “in production”, in order to set up Conductor Live to control the worker nodes and in order to instruct the worker nodes to process video. For background information on these entities, see the [AWS Elemental Conductor Live User Guide](#).

Topics

- [Working with Profiles](#)
- [Working with Channels](#)
- [Channel Scheduling](#)
- [Performing Bulk Tasks on a Channel](#)
- [Controlling Ad Avail on a Channel](#)
- [Working with MPTS](#)
- [Working with Members of an MPTS](#)

Working with Profiles

Topics

- [Recommended Method for Working with Profiles](#)
- [POST: Create a Profile](#)
- [Modify a Profile](#)
- [GET List: Get a List of Profiles](#)
- [GET: Get the Attributes of a Profile](#)
- [DELETE: Delete a Profile](#)

Recommended Method for Working with Profiles

The body of a POST or PUT profile request can contain a lot of elements (attributes of the profile). We recommend that you follow the procedures in this section to create the body.

Create a Profile Base

1. Use the Conductor Live web interface to create a profile that contains most of the content you want:
 - The desired inputs, including the desired number and types of video, audio, and caption streams as well as the desired hot backup fields.
 - The desired output groups.
 - Within each output group, the desired outputs.
 - Within each output, the desired video, audio and captioning.

Give the profile a descriptive name, perhaps including the term “template” and including a description of the inputs, outputs, codecs used, and so on.

See the [AWS Elemental Conductor Live User Guide](#) for information on the contents of a profile.

2. Do a GET Profile List and make a note of the ID for this profile.

Create a Template

1. Use the GET Profile command to get this profile with the clean parameter set to true. For example, to get the profile that has the ID 2, use this command.

```
GET http://198.51.100.0/profiles/2.xml?clean=true
```

The response removes the ID, which makes it valid for re-use in a POST or PUT.

2. Inspect the XML document that is returned. You will notice that it is structured as shown below. Make sure that it has all the information that you want in the template.
3. You can now store this XML as a template and re-use it in the body of a POST.

Re-use the Template

Templates can be used for current version of Conductor Live up to two major versions back. When the profile is uploaded, it is migrated to the current version with field selections and values maintained.

1. To re-use the template, you must:
 - Change the <name>. This name must be unique.

- Change the <permalink>. This must be the <name> converted to lowercase and with spaces converted to underscores. For example, if the <name> is "Profile A", then the <permalink> must be "profile_a".
 - If you are on version 3.2 or higher, you must enter default values for all channel parameters. For more information, see [the section called "POST: Create a Profile"](#).
2. Change any other elements, as desired. Include this XML in the body of a POST or PUT request. For more information, see [the section called "POST: Create a Profile"](#).

XML Structure of a Profile

```
<profile href=> //information about the profile and Conductor Live product and version
  <name>aa</name>
  <permalink>bb</permalink>
  <description>cc</description>
  <input>
    .
    .
    .
  <network_input>
    .
    .
    .
  </network_input>
  <video_selector>    //one or more
    .
    .
    .
  </video_selector>
  <audio_selector>    // one or more
    .
    .
    .
  </audio_selector>
    .
    .
    .
  <caption_selector>  // zero or more
    .
    .
    .
  </caption_selector>
```

```
.  
.   
.   
<stream_assembly>    //one or more  
  <video_description>  
    .  
    .  
    .  
    <h264_settings> //where h264 could be the name of any codec  
    .  
    .  
    .  
    </h264_settings>  
    .  
    .  
    .  
  </video_description>  
  <audio_description>  
    .  
    .  
    .  
    <aac_settings> //where aac could be the name of any audio codec  
    .  
    .  
    .  
    </aac_settings>  
    .  
    .  
    .  
  </audio_description>  
</stream_assembly>  
<output_group>  
  <archive_group_settings>    //where "archive" could be any output group type  
  .  
  .  
  .  
  </archive_group_settings>  
  <output>  
  .  
  .  
  .  
  </output>  
</output_group>
```

```
</profile>
```

POST: Create a Profile

Create a profile.

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/profiles
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

When POSTing xml for a profile originally created with an earlier version of Conductor Live, set content-type to application/vnd.elemental+xml;version=n.n.n, where n.n.n is the number of the Conductor Live version used to create the profile. For example, if the profile was created with Conductor Live version 3.2.1, use:

```
content-type: application/vnd.elemental+xml;version=3.2.1
```

For more information on using profiles created with earlier software versions, see Versioning, [the section called “Header Content for User Authentication”](#).

If you are implementing user authentication, you must also include three authorization headers as described on [the section called “Header Content for User Authentication”](#).

Request Body

The request body contains XML content. For more information, see [the section called “XML Structure of a Profile”](#).

Response

The response repeats back the data that you posted with the addition of:

- id: The newly assigned ID for the group.

The response is identical to the response to a GET Profile. See below for an example.

Example

Response

The response to a valid request presents the new `profile` element with all the information from the new profile.

In this example, the input type is “network_input,” and the URI of that input has been set to a channel parameter “input network location,” and the default value is `udp://255.255.255.255:5001`. For information on channel parameters, see the section on setting up channels in the Conductor Live User Guide. For information on how these parameters are set in a channel, see [the section called “POST: Create a Channel”](#).

```
POST http://198.51.100.0/channels
-----
Content-type:application/vnd.elemental+xml;version=3.2.1
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<profile>
  <id>9</id>
  <name>Profile X</name>
  <permalink>profile_x</permalink>
  <description/>
  <parameters>
    <parameter>
      <name>input network location</name>
      <default>udp://255.255.255.255:5001</default>
    </parameter>
  </parameters>
  <input>
    <network_input>
      <uri>{{input network location}}</uri>
    </network_input>
    <name>input_1</name>
    .
    .
    .
  </input>
  .
  .
```

```
.  
</profile>
```

Creating a Profile for a Channel Used by an MPTS

For important information on the requirements for creating a profile that will be used to create a channel that will become an SPTS in an MPTS output, see [Setting up MPTS outputs](#) in the AWS Elemental Conductor Live User Guide.

Modify a Profile

You cannot modify a profile once it has been created, so there is no PUT command.

GET List: Get a List of Profiles

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/profiles
```

Call Header

- Accept: Set to application/xml

Response

The response contains XML content consisting of one `profiles` element with the following.

- An HREF attribute that specifies the product and version installed on the Conductor Live node.
- Zero or more `profile` elements, one for each profile found. Each element contains several elements, including these key elements. For details on other elements, see [the section called "Recommended Method for Working with Profiles"](#).

Element	Value	Description
id	Integer	The ID for this profile, assigned by the system when the profile is created.

Element	Value	Description
name	String	The name that you assigned to the profile.
permalink	String	The ID of the node, assigned by the system. For profiles, the ID is a string (not an integer) and is created by taking the name you assigned, converting it to lower case, and changing spaces and illegal characters to underscores.

Example

The response to this request specifies that there are three profiles set up in the cluster.

```
<?xml version="1.0" encoding="UTF-8"?>
<profiles href="/profiles" product="AWS Elemental Conductor Live" version="3.3.nnnnn">
  <profile href="/profiles/1">
    <id>1</id>
    <name>Profile A</name>
    <permalink>profile_a</permalink>
    <description></description>
    <input>
      .
      .
      .
    </profile>
  <profile href="/profiles/4">
    <id>4</id>
    <name>Profile C</name>
    <permalink>profile_c</permalink>
    <description></description>
    <input>
      .
      .
      .
    </profile>
  </profiles>
</xml>
```

```
</profile>
<profile href="/profiles/5">
  <id>5</id>
  <name>Profile D</name>
  <permalink>profile_d</permalink>
  <description></description>
  <input>
    .
    .
  </profile>
</profile_list>
```

GET: Get the Attributes of a Profile

Get the attributes of the specified profile.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/profiles/<ID of profile>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#).

Response

The response contains XML content consisting of one profile, with the same elements as the response for GET Profile List, above. For a sketch of XML structure, see [the section called "Recommended Method for Working with Profiles"](#).

Example

This example shows the contents of a profile with the name Profile A, with the permalink "profile_a" (the name in lowercase with spaces replaced by underscores).

```
<?xml version="1.0" encoding="UTF-8"?>
<profile href="/profiles/2" product="AWS Elemental Conductor Live" version="3.0.nnnnn">
```

```
<name>Profile A</name>
<permalink>profile_a</permalink>
<description></description>
.
.
.
</profile>
```

DELETE: Delete a Profile

Delete the specified profile.

You must first verify that the profile is not being used: do a GET Channel List and make sure this profile is not associated with any channel. If it is, do a PUT Channel on that channel so that the channel uses a different profile before you delete this profile.

Request URL

```
DELETE http://<Conductor IP address>/profiles/<ID of profile>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Example

This request deletes the profile with the ID 14.

```
DELETE http://198.51.100.0/profiles/14
```

Working with Channels

Topics

- [POST: Create a Channel](#)
- [PUT: Modify the Attributes of a Channel](#)
- [GET List: Get List of Channels](#)

- [GET: Get the Attributes of a Channel](#)
- [DELETE: Delete a Channel](#)

POST: Create a Channel

Create a new channel.

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/channels
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The request body contains XML content consisting of one `channel` element and some or all of the following elements.

Element	Value	Description
name	String	[Required] A name you assign which must be unique in the cluster.
profile_id	Integer	[Required] The ID of the profile to associate with this channel. To obtain the ID of a specific profile, see the section

Element	Value	Description
		called “GET List: Get a List of Profiles” .
node_id	Integer	[Optional] The ID of the node to associate with this channel. To obtain the ID of a specific node, see the section called “GET List: Get a List of Nodes in the Cluster”.
permalink_name	String	[Optional] A name for the permalink. A permalink provides a mechanism for referencing a channel in a PUT, GET, or DELETE. With a permalink, you can reference a channel immediately after creating it because you already know its value; you don’t have to first do a GET in order to get the automatically assigned ID. If you specify a value in this element, the permalink takes that name. If you leave this element empty, the value is set to be identical to the name element (converted to lower case and with spaces converted to underscores).

Element	Value	Description
channel_params	Array	[Optional] Required if the specified profile contains channel parameters, in which case you must provide values for each of these parameters. Each channel parameter is represented by a name-value pair. Note that the default value set at profile creation is only for profile validation purposes and is not present in the channel. See below for an example.

Response

The response repeats back the data that you posted with the addition of:

- id: element containing the newly assigned ID for the channel.
- status: the current status of the channel. See [the section called “Channel Status Elements”](#) for a list of states.

The response is identical to the response to a GET Channel. See below for an example.

Example

Request

This request creates one channel with the name “Channel C”, associated with the profile that has the ID 3 and the node that that has the ID 10. The profile has a channel parameter called `{{input network location}}`, so the channel must provide a value for that parameter: `udp://239.255.1.10:5001`.

For information on channel parameters, see [Setting Up Channels](#) in the AWS Elemental Conductor Live User Guide. For a list of channel parameters, see [Supported Channel Parameters for Conductor Live](#) in the AWS Elemental Conductor Live User Guide.

```
POST http://198.51.100.0/channels
-----
Content-type:application/xml
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<channel>
  <name>Channel C</name>
  <profile_id>3</profile_id>
  <node_id>10</node_id>
  <permalink_name>Mendis_News_Channel</permalink_name>
  <channel_params type="array">
    <channel_param>
      <name>input network location</name>
      <value>udp://239.255.10.23:5001</value>
    </channel_param>
    <channel_param>
      .
      .
      .
    </channel_param>
  </channel_params>
</channel>
```

Response

In this example, the channel is given the ID 2.

```
<?xml version="1.0" encoding="UTF-8"?>
<channel href="/channels" product="AWS Elemental Conductor Live" version="3.3.0.nnnnn">
  <id>2</id>
  <name>Channel C</name>
  <profile_id>3</profile_id>
  <channel_params type="array">
    <channel_param>
      <name>input network location</name>
      <value>udp://239.255.10.23:5001</value>
    </channel_param>
  </channel_params>
```

```
<node_id>10</node_id>
<permalink_name>Mendis_News_Channel</permalink_name>
<status>idle</status>
</channel>
```

PUT: Modify the Attributes of a Channel

Modify the attributes of the specified channel. The channel must be in a state that allows modifications. To check the status, use GET Channel. For information on the status, see [the section called "Channel Status Elements"](#).

HTTP Request and Response

Request URL

```
PUT http://<Conductor IP address>/channels/<ID of channel>
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#).

Request Body

The body contains only the elements to change; see POST above.

Example

This request changes the name of the channel with the ID 3. It modifies it to use the node that has the ID 1.

```
PUT http://198.51.100.0/channels/3
-----
Content-type:application/xml
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
```

```
<channel>
  <node_id>1</node_id>
</channel>
```

GET List: Get List of Channels

Get a list of all channels, including the attributes of each channel. The attributes include a status.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/channels?metacounts=true
```

where :

- ?metacounts=true is optional.

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response is XML content consisting of one `channels` element with the following.

- An HREF attribute that specifies the product and version installed on the Conductor Live node.
- Zero or more `channel` elements, one for each channel found. Each element contains several elements.

Element	Value	Description
id	Integer	The ID for this channel, assigned by the system when the channel is created.

Element	Value	Description
name	String	The name you assigned to the channel
profile_id	Integer	The ID of the profile associated with this channel.
channel_params	Array	The channel parameters contained in the specified profile, if any.
node_id	Integer	The ID of the node associated with this channel.
status	String	The current status of the channel. See the table below.
permalink_name	String	The name of the permalink associated with this channel.
service_name	String	The service name associated with this channel if the associated profile has the Extract SDT field enabled. Empty if the service name is not part of the source or if the Extract SDT field is disabled.
service_provider_name	String	The service name associated with this channel if the associated profile has the Extract SDT field enabled. Empty if the service provider name is not part of the source or if the Extract SDT field is disabled.

Element	Value	Description
active_alerts	Integer	This setting is included only if the request includes ?metacounts=true. The count of alerts that are currently active for this channel.
recent_error_messages	Integer	This setting is included only if the request includes ?metacounts=true. The count of recent error messages for this channel.

Example

This response shows two channels:

- One channel has the ID 1 and that is associated with profile ID 2 and with node ID 3. This channel also has one channel parameter.
- The other channel has the ID 5 and that is associated with profile ID 4 and with node ID 6. It has no channel parameters.

Also note that the channel with ID 1 has a user-specified permalink while the channel with ID 5 has a permalink that is identical to the channel named: this indicates that the permalink was assigned by the system.

```
GET http://198.51.100.0/channels?metacounts=true
-----
Content-type:application/xml
-----
<channels href="/channels" product="AWS Elemental Conductor Live" version="3.0.nnnnn">
  <channel>
    <id>1</id>
    <name>Channel A</name>
```

```
<profile_id>2</profile_id>
<channel_params type="array">
  <channel_param>
    <name>input network location</name>
    <value>udp://239.255.1.10:5001</value>
  </channel_param>
</channel_params>
<node_id>3</node_id>
<permalink_name>Mendis_News_Channel</permalink_name>
<service_name>MendisNewsChannel</service_name>
<service_provider_name>MendisNetworks</service_provider_name>
<status>running</status>
<active_alerts>2</active_alerts>
<recent_error_messages>0</recent_error_messages>
</channel>
<channel>
  <id>5</id>
  <name>Channel C</name>
  <profile_id>4</profile_id>
  <channel_params type="array"/>
  <node_id>6</node_id>
  <permalink_name>Channel C</permalink_name>
  <service_name>MendisNatureChannel</service_name>
  <service_provider_name>MendisNetworks</service_provider_name>
  <status>idle</status>
  </active_alerts>
  </recent_error_messages>
</channel>
</channel>
</channels>
```

GET: Get the Attributes of a Channel

Get the attributes of the specified channel.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/channels/<ID of channel>?metacounts=true
```

where:

- `?metacounts=true` is optional but adds additional information to the response.

Response

The response contains XML content consisting of one `channel` element with the same elements as the response for GET Channel List, above.

Call Header

- **Accept:** Set to `application/xml`

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#).

Example

This response shows the information for the channel with the ID 1.

```
GET http://198.51.100.0/channels/1
-----
Content-type:application/xml
-----
<channel href="/channels/1" product="AWS Elemental Conductor Live" version="3.3.nnnnn">
  <id>1</id>
  <name>Channel A</name>
  <profile_id>2</profile_id>
  <channel_params type="array">
    <channel_param>
      <name>input network location</name>
      <value>udp://239.255.1.10:5001</value>
    </channel_param>
  </channel_params>
  <node_id>3</node_id>
  <permalink_name>Mendis_News_Channel/permalink_name>
  <service_name>MendisNewsChannel</service_name>
  <service_provider_name>MendisNetworks</service_provider_name>
  <status>running</status>
  <active_alerts>2</active_alerts>
  <recent_error_messages>0</recent_error_messages>
</channel>
```

DELETE: Delete a Channel

Delete the channel that has the specified ID. To get the ID of a specific channel, see [the section called “GET List: Get List of Channels”](#).

The channel must be in a state that allows it to be deleted. To check the status, use GET Channel. For information on the status, see [the section called “Channel Status Elements”](#).

HTTP Request and Response

Request URL

```
DELETE http://<Conductor IP address>/channels/<ID of channel>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Example

This request deletes the channel with the ID 2.

```
DELETE http://198.51.100.0/channels/2
```

Channel Scheduling

Conductor Live schedules channels to run either at a specific time or on a repeating schedule. Specific times are provided via the REST interface within `<run_at>` tags in the format specified in the ISO 8601 standard as combined date and time with UTC offset, such as “2016-07-05T10:09:00-07:00”.

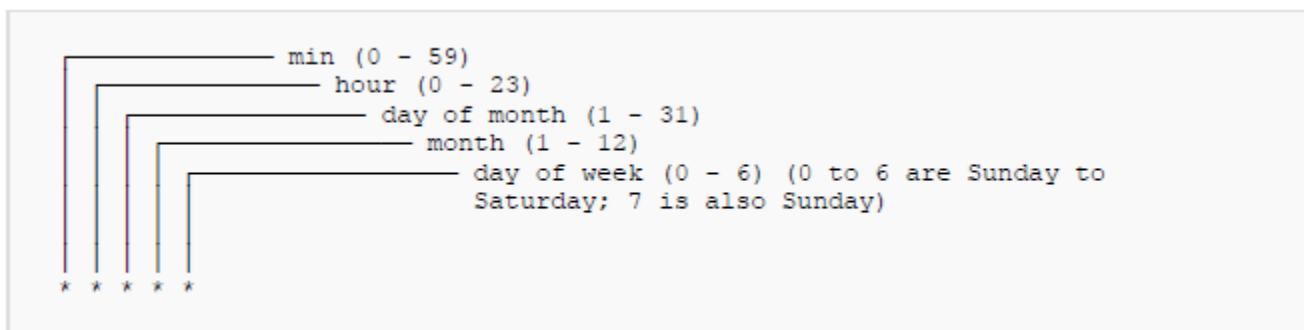
Repeating schedules are provided within `<cron>` tags and represented as CRON elements as specified at crontab.org. The syntax for the schedules is summarized below.

Topics

- [CRON Syntax Summary](#)
- [POST: Create a One-Time Schedule](#)
- [POST: Create a Repeating Schedule](#)
- [POST: Activate a Schedule](#)
- [DELETE: Deactivate a Schedule](#)
- [PUT: Update a Schedule](#)
- [GET List: Get List of All Channel Schedules](#)
- [GET: Get the Attributes of a Schedule](#)
- [GET: Get Schedule Events](#)
- [DELETE: Delete a Schedule](#)

CRON Syntax Summary

The CRON expression is composed of five parts, separated by spaces, representing minute, hour, day of month, month, and day of week respectively, as illustrated below.



The channel runs when the fields in the expression match the current time and date. An asterisk (*) acts as a wildcard and represents all values. If both “day of month” and “day of week” are specified (not *s), then the schedule will run for both.

Each part can specify either one value or a repeating value. A number by itself represents a single value while a number preceded by an asterisk and a forward slash represents a repeating value. For example, in the second position, “5” means “on the fifth hour (5:00 AM)” and “*/5” means “every fifth hour (5:00 AM, 10:00 AM, 3:00 PM, etc).” The table below provides some examples.

Position 1 (min)	Position 2 (hour)	Position 3 (day of month)	Position 4 (month)	Position 5 (day of week)	Resulting Schedule
0	3	*	*	1	Every Monday at 8:00 PM Pacific Daylight Time (UTC-07:00) Note: The hour in position 2 must be provided in UTC, not your local time.
0	3	*	*	1,2,3,4,5	Every weekday at 8:00 PM Pacific Daylight Time (UTC-07:00) Note: The hour in position 2 must be provided in UTC, not your local time.

Position 1 (min)	Position 2 (hour)	Position 3 (day of month)	Position 4 (month)	Position 5 (day of week)	Resulting Schedule
15	1	*	*	1,2,3,4,5	Every weekday at 8:15 PM Eastern Standard Time (UTC-05:00) Note: The hour in position 2 must be provided in UTC, not your local time.
17	*/4	*	*	*	Every four hours on the 17th minute every day. 4:17 AM, 8:17 AM, etc.
0	*/1	*	*	1	Every Monday, each hour at the top of the hour.

POST: Create a One-Time Schedule

Create a schedule that runs only once. Will only run if active. See also [the section called “POST: Create a Repeating Schedule”](#).

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/channels/<channel ID>/schedules
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The request body contains XML content consisting of one schedule element, consisting of some or all of the following elements.

Element	Value Type	Description
name	String	[Required] A name you assign which must be unique in the channel.
active	Boolean	[Optional] If unspecified, defaults to false. A switch indicating whether the schedule will run. “True” for active schedules that run at the appointed time; “false” for inactive schedules that do not run but are saved in the

Element	Value Type	Description
		system and can be activated later.
duration	Integer	[Optional] Length of time in seconds that the channel will run. If unspecified, defaults to nil="true", which results in a schedule that continues to run without stopping.
repeat	Boolean	[Required] The Boolean value for repeat which must be "false" for schedules that run only once.
run_at	Datetime	[Required for one-time schedules] The date and time , in ISO 8601 format, that the schedule begins,in cluding UTC offset. This is present only for schedules that run only once.

Response

The response repeats back the data that you posted with the addition of:

- id: element containing the newly assigned ID for the schedule.
- Any elements not specified in the request body, with default values.

Example

```
POST http://192.0.2.16/channels/1/schedules
-----
Content-type:application/xml
Accept: application/xml
```

```
-----
<?xml version="1.0" encoding="UTF-8"?>
<schedule>
  <name>RunOneHour</name>
  <active>true</active>
  <duration>3600</duration>
  <repeat>false</repeat>
  <run_at>2017-09-06T13:24:00-07:00</run_at>
</schedule>
```

POST: Create a Repeating Schedule

Create a schedule that repeats regularly. Will only run if active. (See also [the section called "POST: Create a One-Time Schedule"](#)).

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/channels/<channel ID>/schedules
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#).

Request Body

XML content consisting of one schedule element, consisting of some or all of the following elements:

Element	Value Type	Description
name	String	[Required] A name you assign which must be unique in the channel.

Element	Value Type	Description
active	Boolean	[Optional] If unspecified, defaults to <code>false</code> . A switch indicating whether the schedule will run. "True" for active schedules that run at the appointed time and <code>false</code> for inactive schedules that do not run but are saved in the system and can be activated later.
duration	Integer	[Required for repeating schedules] The length of time in seconds that the channel will run. If unspecified, defaults to <code>nil="true"</code> , which results in a schedule that continues to run without stopping.
repeat	Boolean	[Required] The Boolean value for repeat which must be "true" for repeating schedules.
cron	CRON expression	[Required for repeating schedules] The expression which specifies the schedule according to the cron standard, as summarized in the section called "CRON Syntax Summary" .

Response

The response repeats back the data that you posted with the addition of:

- `id`: element containing the newly assigned ID for the schedule.
- If not specified in the request, the `active` element, defaulted to “false.”

Examples

Create a schedule repeating every Monday 8:00 PM Pacific Daylight Time (UTC-07:00) for an hour:

```
POST http://192.0.2.16/channels/1/schedules
-----
Content-type:application/xml
Accept: application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<schedule>
  <name>MondaysAtEight</name>
  <active type="boolean">false</active>
  <cron>0 3 * * 1</cron>
  <duration type="integer">3600</duration>
  <repeat type="boolean">true</repeat>
</schedule>
```

Create a schedule repeating every Monday—Friday 8:00 PM Eastern Standard Time (UTC-05:00) for an hour:

```
POST http://192.0.2.16/channels/1/schedules
-----
Content-type:application/xml
Accept: application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<schedule>
  <name>WeekdaysAtEight</name>
  <active type="boolean">false</active>
  <cron>0 1 * * 1,2,3,4,5</cron>
  <duration type="integer">3600</duration>
  <repeat type="boolean">true</repeat>
</schedule>
```

POST: Activate a Schedule

Inactive schedules can be activated as follows.

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/channels/<channel ID>/schedules/<schedule ID>/active
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response shows the schedule element with <active> set to “true” and other elements as described in [the section called “POST: Create a One-Time Schedule”](#) and [the section called “POST: Create a Repeating Schedule”](#). The response also includes the following elements, which are used internally by the system: <args>, <schedulable_type>, and <schedulable_id>. The response is otherwise identical to the response to a GET Schedule. For an example response, see [GET List Example](#).

Example

To activate schedule 17 on channel 1, use this URL.

```
POST http://192.0.2.16/channels/1/schedules/17/active
```

DELETE: Deactivate a Schedule

Active schedules can be deactivated as follows.

HTTP Request and Response

Request URL

```
DELETE http://<Conductor IP address>/channels/<channel ID>/schedules/<schedule ID>/active
```

Call Header

- Accept: application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The system deactivates the schedule but does not return a response. To confirm that the schedule has been deactivated, run a GET on the schedule as described in [the section called “GET: Get the Attributes of a Schedule”](#).

Example

To deactivate schedule 17 on channel 1, use this URL.

```
DELETE http://192.0.2.16/channels/1/schedules/17/active
```

PUT: Update a Schedule

HTTP Request and Response

Request URL

```
PUT http://<Conductor IP Address>/channels/<channel ID>/schedules/<schedule ID>
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

One schedule element that contains the elements to be changed, enclosing the new value.

Response

The system updates the schedule but does not return a response. To confirm that the schedule has been changed, run a GET on the schedule as described in [the section called “GET: Get the Attributes of a Schedule”](#).

Example

To change the name of schedule 17 on channel 1 to “NewName” send the following:

```
PUT http://198.51.100.0/channels/1/schedules/17
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<schedule>
  <name>NewName</name>
</schedule>
```

GET List: Get List of All Channel Schedules

Get a list of active and inactive schedules for a given channel.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP Address>/channels/<channel ID>/schedules
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response contains XML content consisting of one `schedules` element with the following.

- An HREF attribute that specifies the product and version installed on the Conductor Live node.
- Zero or more `schedule` elements, one for each schedule found. Each element contains several elements, as follows.

Element	Value Type	Description
id	Integer	The ID for this schedule, assigned by the system when the schedule is created.
name	String	The name that you assigned to the schedule.
active	Boolean	A switch indicating whether the schedule will run. “True” for active schedules that run at the appointed time and “false” for inactive schedules that do not run but are saved in the system and can be activated later.
duration	Integer	The length of time, in seconds, that the channel will run.
repeat	Boolean	A switch indicating whether the schedule will run more than one time. “True” for repeating schedules and

Element	Value Type	Description
		"false" for schedules that run only once.
run_at	Datetime	The date and time that the schedule begins. This is present only for schedules that run only once.
cron	CRON expression	The expression which specifies the schedule according to the cron standard, as summarized in the section called "CRON Syntax Summary" . This is present only for repeating schedules.

Example

This response shows two schedules:

- One schedule has the ID 13 and is a repeating schedule that runs the channel on weekdays for an hour.
- The other schedule has the ID 20 and will run the channel one time for half an hour.

```
GET http://198.51.100.0/channels/1/schedules
-----
Content-type:application/xml
<?xml version="1.0" encoding="UTF-8"?>
<schedules href="/channels/1/schedules" product="AWS Elemental Conductor Live +
Cable Package + Audio Package + Audio Normalization Package + Audio Decode Package
+ HEVC Package + AWS Elemental Statmux Package + Motion Image Inserter Package"
version="3.2.0.41691" type="array">
  <schedule>
    <id type="integer">13</id>
    <name>MthruF</name>
```

```
<active type="boolean">false</active>
<cron>0 3 * * 1,2,3,4,5</cron>
<duration type="integer">3600</duration>
<repeat type="boolean">true</repeat>
</schedule>
<schedule>
  <id type="integer">20</id>
  <name>OneTime</name>
  <active type="boolean">false</active>
  <duration type="integer">1800</duration>
  <repeat type="boolean">false</repeat>
  <run_at type="datetime">2016-07-07T15:22:00-07:00</run_at>
</schedule>
</schedules>
```

GET: Get the Attributes of a Schedule

Get the details of a specific schedule.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP Address>/channels/<channel ID>/schedules/<schedule ID>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#).

Response

XML content consisting of one schedule element containing the same elements as the response for GET List of All Channel Schedules above.

Example

This response shows the information for the schedule with the ID 13.

```
<schedule>
  <id type="integer">13</id>
  <name>MthruF</name>
  <active type="boolean">false</active>
  <cron>0 3 * * 1,2,3,4,5</cron>
  <duration type="integer">3600</duration>
  <repeat type="boolean">true</repeat>
</schedule>
```

GET: Get Schedule Events

Within the system, active repeating schedules result in “schedule events”. A schedule event is an individual instance of the schedule and represents one time on the calendar when the channel will run. When new schedules are created or updated, and every hour after that, the system generates schedule events to bring the total queued schedule events to 24.

HTTP Request and Response

Request URL

To get all schedule events in the cluster:

```
GET http://<Conductor IP Address>/events
```

To get all schedule events from a single schedule:

```
GET http://<Conductor IP Address>/channels/<channel ID>/schedules/<schedule ID>/events
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response contains XML content consisting of one `schedule_events` element with the following.

- An HREF attribute that specifies the product and version installed on the Conductor Live node.
- Zero or more `schedule_event` elements, one for each schedule event found. Each element contains several elements, as follows.

Element	Value Type	Description
id	Integer	The ID for this schedule event, assigned by the system when the schedule event was generated.
start_at	Datetime	The date and time that the channel will start, in ISO 8601 format, including the UTC offset.
stop_at	Integer	The date and time that the channel will stop, in ISO 8601 format, including the UTC offset.
state	String	An indication of the current state or progress of scheduling. Possible values are: queued, pending, started, stopped, and failed.
message	String	System messages about any unexpected errors which appear here.
schedule_id	Integer	The ID for the schedule that spawned this schedule event.

Example

This response shows the information for the schedule with the ID 15.

```
<?xml version="1.0" encoding="UTF-8"?>
<schedule_events href="/events" product="AWS Elemental Conductor Live + Cable Package
+ Audio Package + Audio Normalization Package + Audio Decode Package + HEVC Package +
AWS Elemental Statmux Package + Motion Image Inserter Package" version="3.2.0.41691"
type="array">
  <schedule_event>
    <id type="integer">801</id>
    <start_at type="datetime">2016-07-08T10:00:00-07:00</start_at>
    <stop_at type="datetime">2016-07-08T10:01:00-07:00</stop_at>
    <state type="integer">1</state>
    <message></message>
    <schedule_id type="integer">15</schedule_id>
  </schedule_event>
</schedule_events>
```

DELETE: Delete a Schedule

Permanently delete a schedule.

HTTP Request and Response

Request URL

```
DELETE http://<Conductor IP Address>/channels/<channel ID>/schedules/<schedule ID>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The system deletes the schedule but does not return a response. To confirm that the schedule has been changed, run a GET on the schedule as described in [the section called “GET: Get the Attributes of a Schedule”](#). The GET should return an error similar to this one:

```
<errors>
```

```
<error>Couldn't find Elemental::Live247::Schedule with id=13 [WHERE  
"live247_schedules"."schedulable_id" = 1 AND "live247_schedules"."schedulable_type" =  
'Elemental::Live247::Channel']</error>  
</errors>
```

Example

This request deletes the schedule with ID 5 on channel 1.

```
DELETE http://198.51.100.0/channels/1/schedules/5
```

Performing Bulk Tasks on a Channel

This section covers commands for the channels entity (starting and stopping channels) and for the task_report entity (monitoring the status of bulk tasks).

Topics

- [POST Start: Start One or More Channels](#)
- [POST Stop: Stop One or More Channels](#)
- [Monitoring Bulk Tasks: GET List of Task Reports](#)
- [Monitoring Bulk Tasks: GET One Task Report](#)

POST Start: Start One or More Channels

Start one or more channels. If the specified channels are currently idle on their nodes, then they will start. If the channels are currently running, they simply continue running.

The channel must be in a state that allows it to be started. To check the status, use GET Channel. For information on the status, see [the section called "Channel Status Elements"](#).

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/channels/start
```

Call Header

- Accept: Set to application/xml

- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The request contains XML content consisting of one `channel_ids` element (of type “array”) with the following.

- One or more `channel_id` elements, one for each channel to start.

Response

The response contains XML content consisting of one `task_report` element, containing the same elements as the response for [GET One Task Report](#). The failed count and success count may both specify 0. The entire element may be missing, indicating that no actions have been performed yet.

Example

Request

This request starts the channels with the IDs 14, 8 and 10.

```
POST http://198.51.100.0/channels/start
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<channel_ids type="array">
  <channel_id>14</channel_id>
  <channel_id>8</channel_id>
  <channel_id>10</channel_id>
</channel_ids>
```

Response

```
<task_report>
  <id>43</id>
```

```
<created_at>2015-05-28T11:29:56-07:00</created_at>
<description>Channel Start</description>
<failed_count>0</failed_count>
<successful_count>0</successful_count>
<task_count>1</task_count>
<updated_at>2015-05-28T11:29:56-07:00</updated_at>
</task_report>
```

POST Stop: Stop One or More Channels

Stop one or more channels. If the specified channels are currently running on their nodes, then they will stop. If the channels are not running, nothing happens.

The channel must be in a state that allows it to be stopped. To check the state and determine its status, use GET Channel. For information on the status, see [the section called “Channel Status Elements”](#).

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/channels/stop/
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The request contains XML content consisting of one `channel_ids` element (of type “array”) with the following.

- One or more `channel_id` elements, one for each channel to stop.

Response

The response contains XML content consisting of one `task_report` element, containing the same elements as the response for [GET One Task Report](#). The failed count and success count may both specify 0. The entire element may be missing, indicating that no actions have been performed yet.

Example

Request

This request starts the channels with the IDs 14 and 10.

```
POST http://198.51.100.0/stop/
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<channel_ids type="array">
  <channel_id>14</channel_id>
  <channel_id>10</channel_id>
</channel_ids>
```

Response

```
<task_report>
  <id>43</id>
  <created_at>2015-05-28T11:29:56-07:00</created_at>
  <description>Channel Start</description>
  <failed_count>0</failed_count>
  <successful_count>0</successful_count>
  <task_count>1</task_count>
  <updated_at>2015-05-28T11:29:56-07:00</updated_at>
</task_report>
```

Monitoring Bulk Tasks: GET List of Task Reports

Get the list of bulk tasks. Each time one of the following commands is performed, a `task_report` is created:

- POST Start Channel
- POST Stop Channel

A task report shows information about the bulk task. The bulk task is made up of individual task items. For example, a POST Start Channel is a bulk task that is made up of one or more task, each to start a different channel.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/task_reports
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response contains XML content consisting of one task_reports element with the following.

- An HREF attribute that specifies the product and version installed on the Conductor Live node.
- Zero or more task_reports elements, one for each bulk task that is in progress or is completed. Each element contains several elements as listed.

Element	Value	Description
id	Integer	The unique ID for this task_report.
created_at	String	The time this bulk task was created (the POST Start or Stop was received by the system). Time is in ISO 8601 format, with the timezone designator indicated as an

Element	Value	Description
		offset from UTC. For example, 2015-08-17T11:59:35-07:00 is the time in the timezone that is 7 hours behind UTC.
updated_at	String	The time the most recent change was made to this task report. In other words, the last time one or more of the count elements was updated.
description	String	"Channel Start" or "Channel Stop".
complete	Boolean	True means the bulk task has completed, with successes and/or failures. False means not all the individual task items have completed yet.
task_count	Integer	The total number of task items in the bulk task. For example, if there are three "start channel" commands, this element specifies 3.
failed_count	Integer	The number of actions that have failed.
successful_count	Integer	The number of actions that have succeeded.

Element	Value	Description
tasks	Array	One or more task elements. Each task represents an individual task within the bulk task. See below.

Recommended Procedure

1. Send a GET task_reports as soon as a bulk task is POSTed.
2. Parse the response for the desired <id> and store that ID for later use. This ID is the ID of the individual bulk task.
3. Parse the response for the complete Boolean .
4. Before that final resolution, you may want to check on the status of individual task items: send a [GET task_reports/<task ID>](#), passing in the ID you saved earlier. Parse the <tasks> array of the response for tasks that are not successful or failed. If necessary for a status of a channel, check the ID within the individual <task> for the channel.

Example

The response to this request provides information on two bulk tasks. One task_report has the ID 4; all of its 10 actions have completed. The other task_report has the ID 5; three of its seven actions have completed.

```
GET http://198.51.100.0/task_reports
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
-----
<?xml version="1.0" encoding="UTF-8"?>
<task_reports>
  <task_report>
    <created_at>2015-04-09T04:12:12-07:00</created_at>
    <description>Channel Start</description>
    <failed_count>1</failed_count>
    <id>4</id>
    <successful_count>9</successful_count>
    <task_count>10</task_count>
    <updated_at>2015-04-09T04:12:13-07:00</updated_at>
```

```
</task_report>
<task_report>
  <created_at>2015-04-09T04:14:12-07:00</created_at>
  <description>Channel Start</description>
  <failed_count>0</failed_count>
  <id>5</id>
  <successful_count>3</successful_count>
  <task_count>7</task_count>
  <updated_at>2015-04-09T04:16:12-07:00</updated_at>
</task_report>
</task_reports>
```

Monitoring Bulk Tasks: GET One Task Report

Get the specified task report related to one bulk task.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/task_reports/<ID of task_report>
```

where <ID of task_report> can be obtained from a GET Task Report List.

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response contains XML content consisting of one task_report element with the following.

- An HREF attribute that specifies the product and version installed on the Conductor Live node.
- Several elements are shown in the table below.

Element	Value	Description
id created_at updated_at description complete task_count failed_count successful_count		The same elements as in the response to GET List of Tasks
tasks	Array	One or more task elements. Each task represents an individual action within the bulk task. See below.

Task Elements

Element	Value	Description
id	Integer	The ID for this task. IDs are unique within the Tasks element.
description	String	The type of bulk task for this task report: "Channel Start" or "Channel Stop."
state	String	The state of this bulk task. When the state changes to "Success," the <successful_count> increments; when

Element	Value	Description
		the state changes to "Failure," then <failure_count> increments.
message	String	"Success" or "Failure."

Example

The response to this request provides information about the task_report with the ID 5; three of its seven actions have completed. The bulk task with ID 3 belongs to Channel 13 and has succeeded, the bulk task with ID 4 belongs to Channel 15 and is pending, and so on.

```
GET http://198.51.100.0/task_reports
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
-----
<?xml version="1.0" encoding="UTF-8"?>
<task_report>
  <created_at>2015-04-09T04:14:12-07:00</created_at>
  <description>Channel Start</description>
  <failed_count>0</failed_count>
  <id>5</id>
  <successful_count>3</successful_count>
  <task_count>7</task_count>
  <updated_at>2015-04-09T04:16:12-07:00</updated_at>
  <tasks type="array">
    <task>
      <id type="integer">3</id>
      <description>Channel Start for Channel 13</description>
      <state>successful</state>
      <message>Success</message>
    </task>
    <task>
      <id type="integer">4</id>
      <description>Channel Start for Channel 51</description>
      <state>pending</state>
      <message>Pending...</message>
    </task>
    <task>
```

```
.  
.   
.   
    </task>  
</tasks>  
</task_report>
```

Controlling Ad Avail on a Channel

This section describes the commands available for controlling ad avail blanking on a channel.

Topics

- [POST Ad Avail State: Start or Stop Ad Avail on a Channel](#)

POST Ad Avail State: Start or Stop Ad Avail on a Channel

Start or stop the ad avail blanking on a channel.

The channel must be running to change the ad avail state. To check the status, use GET Channel. For information on the status, see [the section called “Channel Status Elements”](#).

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/channels/<channel ID>/ad_avail_state/start
```

or

```
POST http://<Conductor IP address>/channels/<channel ID>/ad_avail_state/stop
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

A response is only given if you POST to a channel that is not running or that does not have avail blanking enabled.

Working with MPTS

Topics

- [POST: Create an MPTS](#)
- [PUT: Modify the Attributes of an MPTS](#)
- [GET List: Get a List of MPTS Outputs](#)
- [GET: Get the Attributes of an MPTS Output](#)
- [DELETE: Delete an MPTS Output](#)
- [GET Status List: Get the Status of a List of MPTS Outputs](#)
- [GET Status: Get the Status of an MPTS Output](#)
- [GET Bitrate: Get the Bitrate of an MPTS Output](#)
- [POST Start: Start an MPTS](#)
- [DELETE Stop: Stop an MPTS](#)
- [PUT: Swap Allocation](#)

POST: Create an MPTS

Create an MPTS output. Note that its SPTS programs (members) must be added after the MPTS is created. Use instruction for [POST MPTS member](#).

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/mpts
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The request contains XML content consisting of one **mpts** element, with the following elements.

Element	Value	Description
name	String	A name for the MPTS
node_id	String	The node where the MPTS will be created: where the muxing of the individual programs occurs. Specify a node that is not set up as backup nodes in a redundancy group. To obtain the ID of a specific node, see the section called “GET List: Get a List of Nodes in the Cluster”. Select the correct node. Be aware of the type and choose: - Any AWS Elemental Live node, if you are creating an MPTS consisting only of Constant Bitrate (CBR) programs. - An AWS Elemental Live node with the AWS Elemental Statmux option if you are creating an MPTS that includes at least one AWS Elemental Statmux program.

Element	Value	Description
		An AWS Elemental Statmux stand-alone node if you are creating an MPTS that includes at least one AWS Elemental Statmux program.
permalink_name	String	A name for the permalink . A permalink provides a mechanism for referencing an MPTS in a PUT, GET, or DELETE. With a permalink, you can reference an MPTS immediately after creating it because you already know its value; you don't have to first do a GET in order to get the automatically assigned ID. If you specify a value in this element, the permalink takes that name. If you leave this element empty, the value is set to be identical to the name element (converted to lower case and with spaces converted to underscores).
bitrate	Integer	The total bitrate for the MPTS in bits/second.
video_allocation	Integer	The bitrate to allocate for video traffic in bits/second.

Element	Value	Description
transport_stream_id	Integer	The value for the transport stream ID field in the Program Map Table. Range 0 to 65535.
udp_buffer_size	String	Size of network output buffer. This buffer is used to limit PCR jitter on the network. • Auto: Selects optimal size based on the video elementary stream properties. • Custom: Establishes a non-negative integer for the number of bits to use for the buffer. • Off: Smooths bursts in the network output but does not attempt to limit PCR jitter; this can be used to achieve low-latency streams where output devices do not require buffer-compliant outputs.

Element	Value	Description
output_listening	String	This field is used to set up for output redundancy with output listening and applies only if your Statmux statmux deployment involves AWS Elemental Live nodes as the encoders and an AWS Elemental Statmux node as the muxer; see Setting Up MPTS Outputs in the AWS Elemental Conductor Live User Guide. If you are not setting up for this type of redundancy or if the AWS Elemental Live node is both the encoder and muxer, omit this field.
output_listening_interval	Boolean	The specified detection interval for the output listening feature, in milliseconds. See above for details.

Element	Value	Description
allocation_message_priority	String	This field is used to set up for multiplexer redundancy and applies only if your Statmux deployment involves AWS Elemental Live nodes as the encoders and an AWS Elemental Statmux node as the muxer; see Setting Up MPTS Outputs in the AWS Elemental Conductor Live User Guide. If you are not setting up for this type of redundancy or if the AWS Elemental Live node is both the encoder and muxer, omit this field.
pat_interval	Integer	The PAT interval in ms for the entire MPTS. Range: 10 – 1000 (Default is 100).
destination/uri	String	The primary destination for the MPTS output. This can be a UDP or RTP location. Format: <protocol>://<IP address>:<port>
destination/username	String	The username for the destination, if required.

Element	Value	Description
destination/password	String	The password for the destination, if required.
secondary_destination/ uri	String	Optional. The secondary destination, interface, and virtual source address.
secondary_destination/ username	String	Completing a secondary destination provides “network failure redundancy” for the MPTS. The muxer sends the output to both destination 1 and destination 2; if one destination fails, downstream systems can obtain it from the other. See Setting Up MPTS Outputs in the AWS Elemental Conductor Live User Guide.
secondary_destination/ password	String	If you are not setting up for network failure redundancy, omit these fields.
fec_output_settings	String	See below for details.

Element	Value	Description
additional_system_latency	String	Specify additional time (in milliseconds) to add to the “maximum encoding latency” – the time between when the SPTS channels send their complexity data to the muxer and when the muxer expects the related encoded content for all channels. Typically, specify additional time only if the Buffer Size (set in the video stream) for one of the SPTS channels is longer than typical, or if an SPTS channel is set up for 4 Quadrant-4k HEVC encoding via bonded Live 401 or 402 encoders. Note that you will not be able to change this value while the MPTS is running; you will have to stop the MPTS output and then restart it. This is the only field on the MPTS that you cannot change after creation.
dvb_sdt_settings/rep_interval	Integer	The SDT interval in ms, for the entire MPTS. Range: 25 – 2000 (Default is 500)

Element	Value	Description
dvb_tdt_settings/rep_interval	Integer	The TDT interval in ms, for the entire MPTS. Range: 1000 – 30000 (Default is 1000)
dvb_nit_settings/rep_interval	Integer	The NIT interval in ms, for the entire MPTS. Range: 25 – 10000 (Default is 500)
dvb_nit_settings/network_id	Integer	The numeric identifier of the network to which the MPTS belongs.
dvb_nit_settings/network_name	String	The network name.

fec_output_settings

Element	Value	Description
include_column_fec	Boolean	True means enable column-based FEC; must be true.
include_row_fec	Boolean	True means enables row-based FEC; enabled by default.
column_depth	Integer	Parameter D from SMPTE 2022-1. Range 4-20. The height of the FEC protection matrix. The number of transport stream packets

Element	Value	Description
		per column error correction packet.
row_length	Integer	Parameter L from SMPTE 2022-1. Range 1-20. The width of the FEC protection matrix. This must be between 1 and 20, inclusive . If only Column FEC is used, then larger values increase robustness. If Row FEC is used, then this is the number of transport stream packets per row error correction packet.

Response

The response repeats back the data that you posted, with the addition of:

- id: The newly assigned ID for the MPTS .

The response is identical to the response to a GET MPTS. For a complete example, see [the section called "GET: Get the Attributes of an MPTS Output"](#).

Example

Request

This request creates one MPTS with the name "mpts_A", associated with the node that has the ID 3. The MPTS will contain three channels with channel IDs 3, 6, 7.

```
POST http://198.51.100.0/mpts
-----
<?xml version="1.0" encoding="UTF-8"?>
<mpts>
```

```
<name>mpts_A</name>
<node_id>3</node_id>
<permalink_name>MendisChannelsMPTS</permalink_name>
<bitrate>38800000</bitrate>
<video_allocation>35000000</video_allocation>
<transport_stream_id>1</transport_stream_id>
<udp_buffer_size>Auto</udp_buffer_size>
<output_listening>false</output_listening>
<pat_interval>40</pat_interval>
<destination
  <uri>udp://10.10.10.1:5000</uri>
</destination>
<secondary_destination>
  <uri>udp://10.10.10.40:5000</uri>
</secondary_destination>
<fec_output_settings>
  <include_column_fec>true</include_column_fec>
  <include_row_fec>true</include_row_fec>
  <column_depth>4</column_depth>
  <row_length>6</row_length>
</fec_output_settings>
<allocation_message_priority>primary</allocation_message_priority>
<mpts_members type="array">
  <mpts_member>
    <channel_id>3</channel_id>
    <pid_map>
      <pmt_pid>200</pmt_pid>
      <audio_pids type="array">
        <audio_pid>240</audio_pid>
        <audio_pid>241</audio_pid>
      </audio_pids>
    </pid_map>
    <program_number>1</program_number>
    <type>conductor</type>
  </mpts_member>
  <mpts_member>
    <channel_id>6</channel_id>
    <pid_map>
      <pmt_pid>300</pmt_pid>
      <audio_pids type="array">
        <audio_pid>340</audio_pid>
        <audio_pid>341</audio_pid>
      </audio_pids>
    </pid_map>
```

```
<program_number>2</program_number>
<type>conductor</type>
</mpts_member>
<mpts_member>
  <channel_id>7</channel_id>
  <pid_map>
    <pmt_pid>400</pmt_pid>
    <audio_pids type="array">
      <audio_pid>440</audio_pid>
      <audio_pid>441</audio_pid>
    </audio_pids>
  </pid_map>
  <program_number>3</program_number>
  <type>conductor</type>
</mpts_member>
</mpts_members>
</mpts>
```

Response

The response returns the data you specified, along with:

- A unique `mpts_id` for the MPTS. In this example, the MPTS has been given the ID 12.
- A `created_at` element and `updated_at` element that shows the date and time in ISO 8601 format, with the timezone designator indicated as an offset from UTC. For example, 2015-08-17T11:59:35-07:00 is the time in the time zone that is 7 hours behind UTC.
- An `alerts` element. Typically, this element is empty after initial creation (because the MPTS has not been started).

```
<?xml version="1.0" encoding="UTF-8"?>
<mpts href="/mpts" product="AWS Elemental Conductor Live version="3.3.nnnnn">
  <id>12</id>
  <name>mpts_A</name>
  <node_id>3</node_id>
  .
  .
  .
</mpts>
```

PUT: Modify the Attributes of an MPTS

HTTP Request and Response

Request URL

Change any of the attributes of the specified MPTS output, except the list of SPTS programs. To work with the SPTS programs in the MPTS, use the [MPTS Member commands](#).

```
PUT http://<Conductor IP address>/mpts/<ID of mpts>
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The body contains only the elements to change; see POST above.

Example

This request changes the bitrate and video_allocation of the MPTS with ID 3.

```
PUT http://198.51.100.0/mpts/3
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<mpts>
  <bitrate>30000000</bitrate>
  <video_allocation>25000000</video_allocation>
</mpts>
```

GET List: Get a List of MPTS Outputs

Get the attributes and SPTS programs of all MPTS outputs.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/mpts
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response contains XML content consisting of one `mpts_list` element with the following.

- An HREF attribute that specifies the product and version installed on the Conductor Live node.
- Zero or more `mpts` elements, one for each MPTS found. Each element contains several elements.

Element	Value	Description
id	Integer	The ID for this MPTS output assigned by the system when the MPTS is created.
Other elements	See the section called “POST: Create an MPTS” .	

Example

Request

```
GET http://198.51.100.0/mpts
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<mpts_list href="/mpts_list" product="AWS Elemental Conductor Live"
  version="3.3.nnnnn">
  <mpts>
    <name>mpts_A</name>
    <node_id>3</node_id>
    <permalink_name>MendisChannelsMPTS</permalink_name>
    <bitrate>38800000</bitrate>
    <video_allocation>35000000</video_allocation>
    <transport_stream_id>1</transport_stream_id>
    <udp_buffer_size>Auto</udp_buffer_size>
    <output_listening>false</output_listening>
    <pat_interval>40</pat_interval>
    <destination
      <uri>udp://10.10.10.1:5000</uri>
    </destination>
    <secondary_destination>
      <uri>udp://10.10.10.40:5000</uri>
    </secondary_destination>
    <fec_output_settings>
      <include_column_fec>true</include_column_fec>
      <include_row_fec>true</include_row_fec>
      <column_depth>4</column_depth>
      <row_length>6</row_length>
    </fec_output_settings>
    <additional_system_latency>0</additional_system_latency>
    <allocation_message_priority>primary</allocation_message_priority>
    <alerts type="array">
    <mpts_members type="array">
      <mpts_member>
        <channel_id>3</channel_id>
        <pid_map>
          <pmt_pid>200</pmt_pid>
          <audio_pids type="array">
            <audio_pid>240</audio_pid>
            <audio_pid>241</audio_pid>
          </audio_pids>
        </pid_map>
        <program_number>1</program_number>
        <type>conductor</type>
      </mpts_member>
      <mpts_member>
        <channel_id>6</channel_id>
```

```
<pid_map>
  <pmt_pid>300</pmt_pid>
  <audio_pids type="array">
    <audio_pid>340</audio_pid>
    <audio_pid>341</audio_pid>
  </audio_pids>
</pid_map>
<program_number>2</program_number>
<type>conductor</type>
</mpts_member>
<mpts_member>
  <channel_id>7</channel_id>
  <pid_map>
    <pmt_pid>400</pmt_pid>
    <audio_pids type="array">
      <audio_pid>440</audio_pid>
      <audio_pid>441</audio_pid>
    </audio_pids>
  </pid_map>
  <program_number>3</program_number>
  <type>conductor</type>
</mpts_member>
</mpts_members>
</mpts>
<mpts>
  <mpts_id>
  <name>mpts_B</name>
  <node_id>3</node_id>
  .
  .
  .
  </mpts_members>
</mpts>
<mpts_list>
```

GET: Get the Attributes of an MPTS Output

Get the attributes and SPTS programs of one MPTS output.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/mpts/<ID of mpts>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#) .

Response

The response contains XML content consisting of one mpts element, with the same elements as the response for GET MPTS List above.

DELETE: Delete an MPTS Output

HTTP Request and Response

Request URL

Delete the MPTS output that has the specified ID. To get the ID of a specific MPTS output, see [the section called “GET List: Get a List of MPTS Outputs”](#).

```
DELETE http://<Conductor IP address>/mpts/<id of mpts>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Example

This request deletes the MPTS output with the ID 2.

```
DELETE http://198.51.100.0/mpts/2
```

GET Status List: Get the Status of a List of MPTS Outputs

Get the status of all MPTS outputs.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/mpts/statuses
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#).

Example

This request returns information for the two MPTS outputs that exist in the cluster.

```
GET http://198.51.100.0/mpts/statuses
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
-----
<?xml version="1.0" encoding="UTF-8"?>
<mpts_statuses href="/mpts/statuses" product="AWS Elemental Conductor Live
version="3.3.nnnnn">
  <mpts_status>
    <id type="integer">1</id>
    <status>running</status>
  </mpts_status>
  <mpts_status>
    <id type="integer">4</id>
    <status>running</status>
  </mpts_status>
</mpts_statuses>
```

GET Status: Get the Status of an MPTS Output

HTTP Request and Response

Request URL

Get the status of the specified MPTS output.

```
GET http://<Conductor IP address>/mpts/<ID of mpts>/status
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response contains XML content consisting of one mpts_status element, with the same elements as the response for GET MPTS Status above.

GET Bitrate: Get the Bitrate of an MPTS Output

HTTP Request and Response

Request URL

Get the status of the specified MPTS output.

```
GET http://<Conductor IP address>/mpts/<ID of mpts>/status
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

If the MPTS output has *not* been running in the last hour (meaning no bitrate information exists), then the response is XML content consisting of one `mpts_stats` that contains:

- `id_type`
- an empty `mpts_members` array

If there is bitrate information within the last hour for the MPTS output, then the response is XML content consisting of one `mpts_stats` elements (of type “array”) that contains:

- An HREF attribute that specifies the product and version installed on the Conductor Live node.
- One `mpts_stats` element that contains the following.

Element	Value	Description
<code>mpts_stats</code>		One instance
<code>id</code>	Integer	A temporary ID for the specified MPTS. This ID is ephemeral and should not be stored.
<code>mpts_members</code>	Array	1 or more instances
<code>mpts_member</code>	Array	1 or more instances
<code>id</code>	Integer	A temporary ID for the MPTS member in the specified MPTS. This ID is ephemeral and should not be stored.
<code>name</code>	String	The name of the MPTS member, which is identical to the name attribute of the channel that is associated with this MPTS member. So, if the MPTS member is

Element	Value	Description
		associated with Channel_C , the name of the MPTS member is Channel_C.
series_data		1 instance for each mpts_member.
series_datum		1 or more instances. Each mpts_member contains one series_data that itself contains several series_datum elements. All the MPTS members contain the same number of series_datum. In other words, every member has a snapshot at timestamp "1418939884" (the timestamp may vary by 1 second in different mpts_members).
timestamp	Integer	The timestamp for this piece of data in Unix time.
mpts_id	Integer	Same as the first id: A temporary ID for the specified MPTS.
channel_id	Integer	The ID of the channel that is associated with this MPTS member. So the name (above) and this channel_id refer to the same channel entity.

Element	Value	Description
bitrate	Float	The bitrate (in bits/second) for this MPTS member at the point identified by the timestamp.

Example

This request returns the statistics for the MPTS with the ID 5. This MPTS contains two MPTS members (IDs 30 and 29).

```
GET http://198.51.100.0/mpts/5/stats
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
-----
<?xml version="1.0" encoding="UTF-8"?>
<mpts_stats>
  <id type="integer">10</id>
  <mpts_members type="array">
    <mpts_member>
      <id type="integer">30</id>
      <name>Channel_C</name>
      <series_data type="array">
        <series_datum>
          <timestamp type="integer">1418939884</timestamp>
          <mpts_id type="integer">10</mpts_id>
          <channel_id type="integer">8</channel_id>
          <bitrate type="float">0</bitrate>
        </series_datum>
        <series_datum>
          <timestamp type="integer">1418939886</timestamp>
          <mpts_id type="integer">5</mpts_id>
          <channel_id type="integer">8</channel_id>
          <bitrate type="float">7045000.0</bitrate>
        </series_datum>
        <series_datum>
          .
          .
          .
        </series_datum>
      </series_data>
    </mpts_member>
  </mpts_members>
</mpts_stats>
```

```
</series_data>
</mpts_member>
<mpts_member>
  <id type="integer">29</id>
  <name>Channel_F</name>
  <series_data type="array">
    <series_datum>
      <timestamp type="integer">1418939884</timestamp>
    <series_datum>
      .
      .
      .
    </series_datum>
  </series_data>
</mpts_member>
</mpts_members>
</mpts_stats>
```

POST Start: Start an MPTS

Start an MPTS output. If the specified MPTS output is currently idle, then it will start. If the MPTS output is currently running, it will simply continue running.

When an MPTS output is started, the system looks for the video output from each SPTS program associated with the MPTS output. It looks for this output at the destination specified in that SPTS program. All the SPTS programs are then muxed into one video output.

If there is no output for a given SPTS program, the muxing continues without that SPTS program's content.

If all the SPTS program outputs are missing, the muxing continues without any content. In other words, no content is produced for this MPTS output. Note that a lack of content from all SPTS programs does not cause the MPTS muxing to fail.

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/mpts/<ID of MPTS>/mux
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Example

This request starts the MPTS with the ID 4.

```
POST http://198.51.100.0/mpts/4/mux
```

DELETE Stop: Stop an MPTS

Stop an MPTS output. If the specified MPTS output is currently running, it will stop. If the specified MPTS output is currently idle, it will simply remain idle.

Stopping the MPTS output does not stop the individual SPTS programs associated with the MPTS output.

HTTP Request and Response

Request URL

```
DELETE http://<Conductor IP address>/mpts/<ID of MPTS>/mux
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Example

This request stops the MPTS with the ID 4.

```
DELETE http://198.51.100.0/mpts/4/mux
```

PUT: Swap Allocation

Swap the allocation values (values of the `allocation_message_priority` element) in two related MPTS outputs.

This command is useful when you have set up two MPTS outputs for output listening, in order to support MPTS output redundancy. The two MPTS outputs are a “redundant pair.” This output listening feature is set up as follows:

- Both MPTS outputs are identical with one exception: one of the outputs has its `allocation_message_priority` set to “primary” while the other has it set to “secondary.” Importantly, the two MPTS outputs contain exactly the same SPTS programs and the same destination, so that they are nearly duplicates of each other.
- On the MPTS that is the “secondary”, the `output_listening` element is set to true and the `output_listening_interval` element has a value. In this way, the secondary MPTS output is set up to listen to the primary MPTS output.

This Swap Allocation command lets you switch the roles of the two pairs of MPTS outputs. This switch is achieved by simultaneously setting the `allocation_message_priority` value on the primary output to “secondary” and the value on the secondary output to “primary.”

For this command to succeed, the following conditions must be met:

- The two MPTS outputs must be identical in all values except for the **`allocation_message_priority`** element.
- The `allocation_message_priority` value must be “primary” for one output and “secondary” for the other.
- Every SPTS program in the two MPTS outputs must be used only in these two MPTS outputs.

HTTP Request and Response

Request URL

```
PUT http://<Conductor IP address>/mpts/swap_allocaton_priority
```

Call Header

- Accept: Set to `application/xml`

- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The request body contains XML content consisting of one `mpts_ids` element with the following.

- Two `mpts_id` elements, each containing the ID of an MPTS output.

Example

This request swaps the `allocation_message_priority` values of the MPTS with ID 3 and the MPTS with the ID 4.

```
PUT http://198.51.100.0/mpts/swap_allocation_priority
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<mpts_ids>
  <mpts_id>1</mpts_id>
  <mpts_id>3</mpts_id>
</mpts_ids>
```

Working with Members of an MPTS

Topics

- [POST: Add an SPTS to an MPTS](#)
- [PUT: Modify an SPTS Program](#)
- [PUT: MPTS Channel Swap](#)
- [GET List: Get All SPTS of an MPTS](#)
- [GET: Get an SPTS Program](#)
- [DELETE: Delete an SPTS Program](#)

POST: Add an SPTS to an MPTS

Add an SPTS to the specified MPTS output.

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/mpts/<ID of mpts>/mpts_members
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The request body contains XML content consisting of one `mpts_member` element with the following elements.

Element	Value	Description
type	String	The type is always “conductor.”
program_number	Integer	The program number to use for this SPTS program. 1 – 65535. This must be unique within this MPTS output.
pid_map	See below.	The PID assignments to use for this member in the MPTS output. If no values are provided, PIDs are assigned automatically.

Element	Value	Description
		All PIDs must be unique among all SPTS programs in the MPTS output (not just unique within the individual SPTS program). Support is provided for both PID keys with single values and those with multiple values. See below for details.

Element	Value	Description
channel_id	Integer	The ID of the Conductor Live channel that produces the desired SPTS. To obtain the ID of a specific channel, see the section called “GET List: Get List of Channels”. This channel: • Can belong to a maximum of two different MPTS outputs. • Must include at least one UDP output that is set up to produce input into an MPTS: in other words, the MPTS field is set to Remote (not to Local or None). For information on all the special requirements for the channel, see Create a Profile for MPTS Channels in the AWS Elemental Conductor Live User Guide.

PID Map

The PID map is XML content consisting of one <pid_map> element that contains the following elements.

<pid_map>		0 or 1 instances	Object
	<pmt_pid>	0 or 1 instances	Integer

	<video_pid> <pcr_pid> <scte35_pid> <klv_data_pid> <dvb_tele text_pid> <etv_plat form_pid> <etv_sign al_pid> <timed_me tadata_pid> <private_ metadata_pid> <ecm_pid> <arib_cap tions_pid>			
	<audio_pids>		0 or 1 instances	Object
		<audio_pid>	1 or more instances	Integer
	<dvb_sub_pids>		0 or 1 instances	Object
		<dvb_sub_pid>	1 or more instances	Integer

For example:

<pid_map>

```
<pmt_pid>888</pmt_pid>
<audio_pids type="array">
  <audio_pid>240</audio_pid>
  <audio_pid>241</audio_pid>
</audio_pids>
</pid_map>
```

Response

The response repeats back the data that you posted with the addition of:

- id: The newly assigned ID for the mpts_member.

The response is identical to the response to a GET MPTS Member. See below for an example.

Example

This example shows the result of adding the channel with the ID 2 as an SPTS program with the program ID 5 in the MPTS output that has the ID 3.

Request

```
POST http://198.51.100.0/mpts/3/mpts_members
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<mpts_member>
  <type>conductor</type>
  <program_number>5</program_number>
  <channel_id>2</channel_id>
  <pid_map>
    <pmt_pid>400</pmt_pid>
    <audio_pids type="array">
      <audio_pid>440</audio_pid>
      <audio_pid>441</audio_pid>
    </audio_pids>
  </pid_map>
</mpts_member>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<mpts_member href="/mpts/3/mpts_member/4" product="AWS Elemental Conductor Live"
version="1.0.3b.12345>
  <id>4</id>
  <channel_id>2</channel_id>
  <pid_map>
    <pmt_pid>400</pmt_pid>
    <audio_pids type="array">
      <audio_pid>440</audio_pid>
      <audio_pid>441</audio_pid>
    </audio_pids>
  </pid_map>
  <program_number>5</program_number>
  <type>conductor</type>
</mpts_member>
```

PUT: Modify an SPTS Program

Change the PID map and program number of the specified MPTS member.

Do not use PUT to modify the channel ID; instead, delete the MPTS member (DELETE) and recreate it (POST) with the new channel ID.

HTTP Request and Response

Request URL

```
PUT http://<Conductor IP address>/mpts/<ID of mpts>/mpts_members/<ID of mpts member>
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The body contains only the elements to change; see POST above.

Example

This example changes the MPTS member with the ID 6 in the MPTS that has the ID 2. The request changes one of the PID values.

```
PUT http://198.51.100.0/mpts/2/mpts_members/6
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<mpts_member>
  <pid_map>
    <pmt_pid>600</pmt_pid>
  </pid_map>
</mpts_member>
```

PUT: MPTS Channel Swap

Swap the channel assigned to the MPTS.

HTTP Request and Response

Request URL

```
PUT http://<Conductor IP address>/mpts/<ID of mpts>
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The request body contains XML content consisting of one `mpts_members` element, consisting of the following.

- Two `mpts_member` elements:

- One to remove the existing channel assignment, containing the following elements.

Element	Value	Description
program_number	Integer	The program number used for this SPTS program. To obtain the program number for a specific channel, see the section called “GET List: Get All SPTS of an MPTS” .
pid_map	See the PID map .	PID assignments to use for this member in the MPTS output. If no values are provided, PIDs will be assigned automatically. All PIDs must be unique among all SPTS programs in the MPTS output (not just unique within the individual SPTS program). Support is provided for both PID keys with single values and those with multiple values. See below for details.

Element	Value	Description
channel_id	integer	The ID of the new Conductor Live channel that will produce the desired SPTS. To obtain the ID of a specific channel, see the section called “GET List: Get List of Channels”. This channel: • Can belong to a maximum of two different MPTS outputs. • Must include at least one UDP output that is set up to produce input into an MPTS: in other words, the MPTS field is set to Remote (not to Local or None). For information on all the special requirements for the channel, see Create a Profile for MPTS Channels in the AWS Elemental Conductor Live User Guide.

- Another to add a new channel assignment, using the following elements.

Element	Value	Description
type	String	Always “conductor”

Element	Value	Description
program_number	Integer	The program number used for this SPTS program. To obtain the program number for a specific channel, see the section called “GET List: Get All SPTS of an MPTS” .
pid_map	See the PID map .	The PID assignments to use for this member in the MPTS output. If no values are provided, PIDs are assigned automatically. All PIDs must be unique among all SPTS programs in the MPTS output (not just unique within the individual SPTS program). Support is provided for both PID keys with single values and with multiple values. See below for details.

Element	Value	Description
channel_id	Integer	The ID of the new Conductor Live channel that produces the desired SPTS. To obtain the ID of a specific channel, see the section called “GET List: Get List of Channels”. This channel: • Can belong to a maximum of two different MPTS outputs. • Must include at least one UDP output that is set up to produce input into an MPTS: in other words, the MPTS field is set to Remote (not to Local or None). For information on all the special requirements for the channel, see the section on creating a profile for MPTS in the AWS Elemental Conductor Live User Guide.

Example

This request swaps channel 3 for channel 4 and uses default PID value (so does not specify any PIDs).

Request

```
PUT http://198.51.100.0/mpts/1
```

```

-----
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<mpts>
  <mpts_members type="array">
    <mpts_member>
      <channel_id>3</channel_id>
      <_destroy>1</_destroy>
    </mpts_member>
    <mpts_member>
      <type>conductor</type>
      <program_number>1</program_number>
      <channel_id>4</channel_id>
    </mpts_member>
  </mpts_members>
</mpts>

```

Response

PUT http://198.51.100.0/mpts/1

```

-----
<?xml version="1.0" encoding="UTF-8"?>
<mpts>
  <id type="integer">1</id>
  <node_id type="integer">2</node_id>
  <created_at type="datetime">2016-05-18T07:54:06-07:00</created_at>
  <updated_at type="datetime">2016-05-18T07:54:06-07:00</updated_at>
  <additional_system_latency type="integer">0</additional_system_latency>
  <allocation_message_priority>primary</allocation_message_priority>
  <bitrate type="integer">38800000</bitrate>
  <name>mpts</name>
  <output_listening type="boolean">>false</output_listening>
  <controller_only type="boolean">>false</controller_only>
  <output_listening_interval type="integer">500</output_listening_interval>
  <pat_interval type="integer">100</pat_interval>
  <transport_stream_id type="integer">1</transport_stream_id>
  <udp_buffer_size>Auto</udp_buffer_size>
  <video_allocation type="integer">35000000</video_allocation>
  <permalink_name>mpts</permalink_name>
  <mpts_members type="array">
    <mpts_member>

```

```
<id type="integer">2</id>
<mpts_id type="integer">1</mpts_id>
<created_at type="datetime">2016-05-18T08:06:26-07:00</created_at>
<updated_at type="datetime">2016-05-18T08:06:27-07:00</updated_at>
<channel_id type="integer">4</channel_id>
<name>mpts channel - duplicate</name>
<pid_map>
  <pmt_pid type="integer">100</pmt_pid>
  <video_pid type="integer">101</video_pid>
  <dvb_teletext_pid type="integer">105</dvb_teletext_pid>
  <audio_pids type="array">
    <audio_pid type="integer">140</audio_pid>
    <audio_pid type="integer">141</audio_pid>
    <audio_pid type="integer">142</audio_pid>
    <audio_pid type="integer">143</audio_pid>
    <audio_pid type="integer">144</audio_pid>
    <audio_pid type="integer">145</audio_pid>
    <audio_pid type="integer">146</audio_pid>
    <audio_pid type="integer">147</audio_pid>
    <audio_pid type="integer">148</audio_pid>
    <audio_pid type="integer">149</audio_pid>
    <audio_pid type="integer">150</audio_pid>
    <audio_pid type="integer">151</audio_pid>
    <audio_pid type="integer">152</audio_pid>
    <audio_pid type="integer">153</audio_pid>
    <audio_pid type="integer">154</audio_pid>
    <audio_pid type="integer">155</audio_pid>
    <audio_pid type="integer">156</audio_pid>
  </audio_pids>
  <ecm_pid type="integer">110</ecm_pid>
  <arib_captions_pid type="integer">111</arib_captions_pid>
</pid_map>
<program_number type="integer">1</program_number>
<type>conductor</type>
<input>
  <uri>rtp://239.20.20.20:5001</uri>
</input>
<secondary_input>
  <uri>rtp://239.20.20.20:5010</uri>
</secondary_input>
<allocation_transmit_destination>
  <uri>udp://239.20.30.20:5001</uri>
</allocation_transmit_destination>
<secondary_allocation_transmit_destination>
```

```
<uri>udp://239.20.20.80:5001</uri>
</secondary_allocation_transmit_destination>
<complexity_receipt_destination>
  <uri>udp://239.10.20.20:5001</uri>
</complexity_receipt_destination>
<secondary_complexity_receipt_destination>
  <uri>udp://239.20.40.20:5001</uri>
</secondary_complexity_receipt_destination>
</mpts_member>
</mpts_members>
<destination>
  <uri>udp://10.10.10.1:5000</uri>
</destination>
<secondary_destination nil="true"/>
<fec_output_settings nil="true"/>
<dvb_sdt_settings nil="true"/>
<dvb_tdt_settings nil="true"/>
<dvb_nit_settings nil="true"/>
</mpts>
```

GET List: Get All SPTS of an MPTS

Get a list of all the SPTS programs in the specified MPTS output.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/mpts/<ID of mpts>/mpts_members
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#).

Response

The response contains XML content consisting of one mpts_members element with the following.

- An HREF attribute that specifies the product and version installed on the Conductor Live node.

- Zero or more `mpts_member` elements, one for each mpts member found. Each `mpts_member` element contains several elements.

Element	Value	Description
id	Integer	The ID for this MPTS member, assigned by the system when the MPTS member is created (either as part of the creation of an MPTS, or when creating an individual MPTS member).
mpts_id	Integer	
created_at	String	The time this bulk task was created (the POST Start or Stop was received by the system). Time is in ISO 8601 format, with the timezone designator indicated as an offset from UTC. For example, 2015-08-17T11:59:35-07:00 is the time in the timezone that is 7 hours behind UTC.
updated_at	String	The time the most recent change was made to this task report, in other words, the last time one or more of the count elements was updated.
name	String	The name (value of the name element) of the channel that is a member of the MPTS. You did not specify this name

Element	Value	Description
		in the POST MPTS member; instead, the system finds it and includes it in the GET response.
Elements created by the POST MPTS Member	See the section called “POST: Create an MPTS” .	

Element	Value	Description
input	String	This is identical to the primary destination that you specified in the profile used to create this channel (which is now an mpts_member). This primary destination is specified in the profile XML by the following: output_group/output/udp_settings/destination/uri output_group/output/udp_settings/destination/interface output_group/output/udp_settings/destination/username output_group/output/udp_settings/destination/password output_group/output/udp_settings/destination/certificate file Keep in mind that the <i>destination</i> for the profile (channel) becomes <i>input</i> for the MPTS.

Element	Value	Description
secondary_input	String	This is identical to the secondary destination that you specified in the profile used to create this channel (which is now an mpts_member). It may be blank. This secondary destination is specified in the profile XML by the following tags: output_group/output/udp_settings/secondary_destination/uri output_group/output/udp_settings/secondary_destination/interface output_group/output/udp_settings/secondary_destination/username output_group/output/udp_settings/secondary_destination/password output_group/output/udp_settings/secondary_destination/certificate file Keep in mind that the <i>destination</i> for the profile (channel) becomes <i>input</i> for the MPTS.

Element	Value	Description
allocation_transmit_destination	String	This is identical to the following tag that you included in the profile used to create this channel (which is now an mpts_member): output_group/output/udp_settings/allocation_receipt_destination Keep in mind that the <i>receipt</i> destination for the profile (channel) becomes a <i>transmit</i> destination for the MPTS.
complexity_receipt_destination	String	This is identical to the following tag that you included in the profile used to create this channel (which is now an mpts_member): output_group/output/udp_settings/complexity_transmit_destination Keep in mind that the <i>transmit</i> destination for the profile (channel) becomes a <i>receipt</i> destination for the MPTS.

Element	Value	Description
secondary_allocation_transmit_destination	String	This is identical to the following tag that you included in the profile used to create this channel (which is now an mpts_member): output_group/output/udp_settings/secondary_allocation_receipt_destination Keep in mind that the <i>receipt</i> destination for the profile (channel) becomes a <i>transmit</i> destination for the MPTS.
secondary_complexity_receipt_destination	String	This is identical to the following tag that you included in the profile used to create this channel (which is now an mpts_member): output_group/output/udp_settings/secondary_complexity_transmit_destination Keep in mind that the <i>transmit</i> destination for the profile (channel) becomes a <i>receipt</i> destination for the MPTS.

Example

Request

```
GET http://198.51.100.0/mpts/3/mpts_members
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<mpts_members>
  <mpts_member>
    <id type="integer">3</id>
    <mpts_id type="integer">3</mpts_id>
    <created_at type="datetime">2015-08-18T13:46:23-07:00</created_at>
    <updated_at type="datetime">2015-08-18T13:46:23-07:00</updated_at>
    <channel_id type="integer">6</channel_id>
    <name>Channel_A</name>
    <pid_map>
      <pmt_pid type="integer">100</pmt_pid>
      <video_pid type="integer">101</video_pid>
    </pid_map>
    <program_number type="integer">1</program_number>
    <type>conductor</type>
    <input>
      <uri>udp://127.0.0.1:5004</uri>
    </input>
    <secondary_input nil="true"/>
    <allocation_transmit_destination>
      <uri>udp://127.0.0.1:5005</uri>
    </allocation_transmit_destination>
    <secondary_allocation_transmit_destination nil="true"/>
    <complexity_receipt_destination>
      <uri>udp://127.0.0.1:5002</uri>
    </complexity_receipt_destination>
    <secondary_complexity_receipt_destination nil="true"/>
  </mpts_member>
</mpts_members>
```

GET: Get an SPTS Program

Get the specified SPTS program from the specified MPTS output.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/mpts/<ID of mpts>/mpts_members/<ID of mpts member>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#).

Response

The response contains XML content consisting of one **mpts_member** element, with the same elements as the response for GET SPTS Program List above.

Example

TEXT

DELETE: Delete an SPTS Program

Delete the specified SPTS program from the specified MPTS output.

HTTP Request and Response

Request URL

```
DELETE http://<Conductor IP address>/mpts/mpts_id/mpts_members/<mpts_member_id>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#).

Example

Delete the SPTS program with the ID 4 from the MPTS with the ID 2.

```
DELETE http://198.51.100.0/mpts/2/mpts_members/4
```

Monitoring Conductor Live

The entities in this chapter are those that you work with “in production”, in order to set up Conductor Live to control the worker nodes and in order to instruct the worker nodes to process video. For background information on these entities, see the [AWS Elemental Conductor Live User Guide](#).

Topics

- [Managing Channels](#)
- [Querying Alerts and Messages](#)

Managing Channels

Channel Status Elements

The status element of a channel provides information about the current status of the channel. The status imposes rules about the ability to modify (PUT), delete (DELETE), or start or stop (POST Start Channel and POST Stop Channel) a channel.

You can obtain the status of a channel using [GET Channel List](#) or [GET Channel](#).

Status	Modify	Delete	Start	Stop
Active	Not allowed	Not allowed	Not allowed	Allowed
Complete	Allowed	Allowed	Allowed	Not applicable
Error	Allowed. Status will change to idle.	Allowed	Not allowed	Not applicable
Idle	Allowed	Allowed	Allowed	Not applicable
Pending	Not allowed	Not allowed	Not allowed	Not allowed
Postprocessing	Not allowed	Not allowed	Not allowed	Allowed
Preprocessing	Not allowed	Not allowed	Not allowed	Allowed

Status	Modify	Delete	Start	Stop
Running	Not allowed	Not allowed	Not allowed	Allowed
Started	Not allowed	Not allowed	Not allowed	Allowed
Starting	Not allowed	Not allowed	Not allowed	Allowed
Stopping	Not allowed	Not allowed	Not allowed	Not applicable
Suspended	Not allowed	Not allowed	Not allowed	Allowed
Unknown	Not allowed	Not allowed	Not allowed	Not allowed
Unreachable	Not allowed	Not allowed	Not allowed	Not allowed

POST Channel Revert: Resetting the Channel

If a channel is in one of the following statuses, you can make it revert to the Idle state:

- Complete
- Error
- Unknown

If you try to revert to Idle from another status, the system ignores the command.

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request URL

```
POST http://<Conductor IP address>/channels/<ID of channel>/revert
```

Querying Alerts and Messages

GET Alerts: Get a List of Alerts

Get a list of alerts on one or more nodes in the cluster.

HTTP Request and Response

Request URL

The request URL consists of the operation, the URL of the Conductor Live node, the alerts request, and optional filter parameters. Use this format:

```
GET http://<Conductor IP address>/alerts?<filter>=<value>&<filter>=<value>
```

The following filter parameters are available for the GET request. See the [examples](#), which show a variety of filter parameters.

 **Note**

Requests for alert data can return very large amounts of information. To receive that information in manageable chunks, use pagination as described in [the section called “Specifying Pagination of the Response”](#).

Filter	Value	Description and Notes
status	String	Filter for only those alerts that are “active” or “inactive”. To show all active alerts, use: GET http://198.51.100.0/alerts?status=active To show all inactive alerts, use:

Filter	Value	Description and Notes
		GET http://198.51.100.0/alerts?status=inactive In both examples above, the Conductor IP address is 198.51.100.0.
origin	Integer	Filter for alerts that originate from a specific node. See the section called “Where Alerts Come From and Where They Apply” for details. GET http://198.51.100.0/alerts?origin=2 Use the IP address of the Conductor node and the ID of the node as assigned by Conductor Live. In the example above, the Conductor IP address is 198.51.100.0 and the originating node ID is 2. To determine the node ID, send a GET request.

Filter	Value	Description and Notes
node	Integer	Filter for alerts that apply to a specific node. See the section called “Where Alerts Come From and Where They Apply” for details. Typically, you do not filter for both origin and node. <pre>GET http://198.51.100.0/alerts?node=2</pre> In the example above, the Conductor Live IP address is 198.51.100.0 and the originating node ID is 2. To determine the node ID, send a GET request.

Filter	Value	Description and Notes
channel	Integer	Filter for alerts that apply to a specific channel. <pre>GET http://198.51.100.0/alerts?channel=4</pre> In the example above, the Conductor Live IP address is 198.51.100.0 and the ID of the channel the alert is coming from is 4. To determine the channel ID, send a GET request as shown in this example. To get all alerts originating from a specific node that apply to a specific channel, you can filter for origin and channel together: <pre>GET http://198.51.100.0/alerts?channel=4&origin=2</pre>

Filter	Value	Description and Notes
mpts	Integer	Filter for alerts that apply to a specific Multi-Program Transport Stream (MPTS) output. <pre>GET http://198.51.100.0/alerts?mpts=3</pre> Enter the ID of the MPTS output as assigned by Conductor Live. You can filter for both an origin and an MPTS.
type	String	Filter for alerts by alert type: CpuAlert, NodeStatusAlert, DiskAlert, InputAlert, and so on. All alerts types that appear in the user interface under Status>Alerts are valid type values. <pre>GET http://198.51.100.0/alerts?type=DiskAlert</pre> In the example above, the Conductor Live IP address is 198.51.100.0.

Filter	Value	Description and Notes
code	String	Filter for only those alerts with the specified alert codes. Enter a list of 1 or more numeric codes. All alerts codes that appear in the user interface under Status>Alerts are valid code values. <pre>GET http://198.51.100.0/alerts?code=4002</pre> In the example above, the Conductor Live IP address is 198.51.100.0 and 4002 is the code for DiskAlert.

Where Alerts Come From and Where They Apply

An alert may originate with an Conductor Live node or it may originate with a worker node and get pushed to Conductor Live. In both cases, the alert may relate to a node and/or a channel or MPTS output.

For example, if a channel on Live node live-01 fails, then an alert occurs. The alert originates from Live node live-01 and applies to node live-01 and channel Channel_A.

Or a node live-01 may fail, resulting in an alert. In this case, the alert originates from the Conductor node (not from live-01), but it applies to node live-01.

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#) .

Response

The response is XML content consisting of one **alerts** element that holds the following:

- An HREF attribute that specifies the product and version installed on the Conductor Live node.
- Zero or more `alert` elements each containing several elements from this table.

Element	Value	Description
alertable_id	Integer	The unique ID for the entity in alertable_type. This is the ID that is included in the response of a GET. For example, GET Node includes the ID that uniquely identifies the node.
alertable_type	String	The entity that the alert is associated with: Node, Channel, MPTS.
eme_id	Integer	An internal ID.
id	Integer	A unique ID for this alert, assigned by Conductor Live.
last_set	Time	The time that the alert was last set (either when it appeared for the first time or when it changed back to set after previously being cleared). Some alerts are set and cleared repeatedly, so the time for these alerts may change.

Element	Value	Description
		Some are not set and cleared repeatedly, so the time will never change.
message	String	A longer description.
name	String	A short description.
node_id	Integer	The ID that identifies the same piece of data as the origin filter in the request; see the section called “GET Alerts: Get a List of Alerts” .
notes	String	A section that shows whether the web interface user has added a note to the alert (Status > Notifications): it appears here.
output_id	Integer	The unique ID for the entity in output_type.
output_type	String	The type of the output in the channel (event) that this alert specifically applies to. Otherwise, null.
quiet	Boolean	The status of suppression of the alert on the worker node. True means the alert has been suppressed on the worker node. False means it has not been suppressed.

Element	Value	Description
remote_id	Integer	The ID of the entity (channel or MPTS) as assigned by the worker node that is running this entity. The ID assigned by Conductor Live may be different from the ID assigned by the worker node.
set	Boolean	The status of whether the alert has been set. True means it is set. False means it has been cleared or has never been set.
threshold		The indicator which applies only to alerts that have a threshold assigned to the alerting context. For example, this can be the CPU Alert, which specifies when CPU usage is above a given percentage. If the alert is the type of alert that gets set repeatedly, each time the threshold changes, then this number shows the minimum threshold. The alert does not get set for this first time until this threshold is met.
type	String	The type of the alert.
updated_at	Time	The last time this alert was updated.

Examples

Example 1

This example requests all active alerts from node 2.

```
GET http://10.4.138.230/alerts?channel=5&status=active
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
-----
<?xml version="1.0" encoding="UTF-8"?>
<alerts href="/alerts" product="AWS Elemental Conductor Live" version="3.2.0.40946"
type="array">
  <alert>
    <id type="integer">18</id>
    <type>InputAlert</type>
    <name>Live Event 125 Alert 5302</name>
    <message>Stopped receiving network data on [udp://239.255.10.7:5001]</message>
    <last_set type="datetime">2016-06-01T15:52:17-07:00</last_set>
    <quiet type="boolean">false</quiet>
    <set type="boolean">true</set>
    <threshold nil="true"/>
    <code type="integer">5302</code>
    <notes nil="true"/>
    <alertable_type>Elemental::Live247::Channel</alertable_type>
    <alertable_id type="integer">3</alertable_id>
    <node_id type="integer">2</node_id>
    <updated_at type="datetime">2016-06-01T15:52:18-07:00</updated_at>
    <readable_type>Channel</readable_type>
    <product_type nil="true"/>
  </alert>
  <alert>
    <id type="integer">16</id>
    <type>InputAlert</type>
    <name>Live Event 125 Alert 5201</name>
    <message>[188] Video not detected: Check input signal</message>
    <last_set type="datetime">2016-06-01T15:52:21-07:00</last_set>
    <quiet type="boolean">false</quiet>
    <set type="boolean">true</set>
    <threshold nil="true"/>
    <code type="integer">5201</code>
    <notes nil="true"/>
    <alertable_type>Elemental::Live247::Channel</alertable_type>
    <alertable_id type="integer">3</alertable_id>
```

```
<node_id type="integer">2</node_id>
<updated_at type="datetime">2016-06-01T15:52:21-07:00</updated_at>
<readable_type>Channel</readable_type>
<product_type nil="true"/>
</alert>
</alerts>
```

Example 2

The following example requests all *active input* alerts on *channel 5* but limits the number to *10 alerts per page*.

```
GET http://10.4.138.230/alerts?channel=5&type=InputAlert&status=active&per_page=10
```

GET Messages: Get a List of Messages

Get a list of messages on one or more nodes in the cluster.

HTTP Request and Response

Request URL

The request consists of the operation, the URL of the Conductor Live node, the messages request, and optional filter parameters. If using multiple filters, enter the ampersand (&) between each filter parameter. For example:

```
GET http://<Conductor IP address>/messages?<filter>=<value>&<filter>=<value>
```

The following filter parameters are available for the GET request. See the [examples](#) for a variety of filter parameters.

Filter	Value	Description
origin	Integer	Filter for messages that were originate with a specific node. Enter the ID of the node as assigned by Conductor Live.

Filter	Value	Description
		See the section called “Where Alerts Come From and Where They Apply” .
node	Integer	Filter for messages that apply to a specific node. Enter the ID of the node as assigned by AWS Elemental Conductor Live 3. Typically, for a node, do not enter both the origin and node filters.
channel	Integer	Filter for messages that apply to a specific channel. Enter the ID of the channel as assigned by Conductor Live. You can enter both an origin and channel.
mpts	Integer	Filter for messages that apply to a specific Multi-Program Transport Stream (MPTS) output. Enter the ID of the MPTS output as assigned by Conductor Live. You can enter both an origin and an MPTS.
type	String	Case-insensitive match. See the Status > Messages screen on the Conductor Live web interface for a list of applicable types.

Filter	Value	Description
code	String	Filter for only those messages with the specified message codes. Enter a comma-separated list of 1 or more numeric codes. For a list of codes, see the Status > Messages screen on the Conductor Live web interface.
page	Integer	Default is 1.
per_page	Integer	Default is 20.

Origin, Node, Channel, MPTS

A message may originate with an Conductor Live node or it may originate with a worker node and get pushed to Conductor Live. In both cases, the message may relate to a node and/or a channel or Multi-Program Transport Stream (MPTS) output.

For example, if a channel on Live node live-01 fails, then a message may occur. The message originates from Live node live-01 and applies to node live-01 and channel Channel_A.

Or a node live-01 may fail, resulting in a message. In this case, the message originates from the Conductor node (not from live-01), but it applies to node live-01.

page and per_page

The per_page parameter chunks messages into groups or “pages” of a given count. For example, “30” means chunk messages into one page for IDs 1-30, another for 31-60, and so on. The page parameter identifies a given page. For example, if per_page is “30,” then page 3 contains IDs 61-90.

Enter only the page (per_page uses the default), only the per_page (page uses the default), or both.

Messages are not sorted into any order, so you cannot obtain the newest messages, for example. The key use for these filter parameters is to retrieve a large number of messages by specifying a high number in per_page.

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#) .

Response

The response is XML content consisting of one **message** element with the following.

- An HREF attribute that specifies the product and version installed on the Conductor Live node.
- Several elements from the table below.

Element	Value	Description
id	Integer	A unique ID for this alert, assigned by Conductor Live.
type	String	The type of the message.
code	Integer	The internal code assigned to this type of alert.
message	String	A longer description.
data	String	A data string that can hold additional information on the message.
notes	String	A notes string that is always null.
messageable_id	Integer	The unique ID for the entity in alertable_type. This is the ID that is included in the response of a GET. For example, GET Node includes

Element	Value	Description
		the ID that uniquely identifies the node.
messageable_type	String	The type of the output in the channel (event) that this alert specifically applies to. Otherwise null.
node_id	Integer	The ID which identifies the same piece of data as the origin filter in the request; see the section called “GET Alerts: Get a List of Alerts” .
remote_id	Integer	The ID of the entity (channel or MPTS) as assigned by the worker node that is running this entity. The ID assigned by Conductor Live may be different from the ID assigned by the worker node.
updated_at	Time	The last time this alert was updated.

Examples

Example 1

This example requests all active messages for node 2.

```
GET http://10.4.138.230/messages?node=2&status=active
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
-----
<?xml version="1.0" encoding="UTF-8"?>
```

```
<messages href="/messages" product="AWS Elemental Conductor Live" version="3.3.nnnnn"
  type="array">
<messages>
  <message>
    <id type="integer">5</id>
    <type>AuditMessage</type>
    <code type="integer">30</code>
    <message>Node live_1 activated</message>
    <data nil="true"/>
    <notes nil="true"/>
    <messageable_type>Elemental::Live247::Node</messageable_type>
    <messageable_id type="integer">2</messageable_id>
    <remote_id type="integer">280</remote_id>
    <node_id type="integer">2</node_id>
    <updated_at type="datetime">2016-04-28T09:40:18-07:00</updated_at>
    <readable_type>Node</readable_type>
    <ignore type="boolean">false</ignore>
  </message>
  <message>
    <id type="integer">6</id>
    <type>AuditMessage</type>
    <code type="integer">32</code>
    <message>Node live_1 added to cluster</message>
    <data nil="true"/>
    <notes nil="true"/>
    <messageable_type>Elemental::Live247::Node</messageable_type>
    <messageable_id type="integer">2</messageable_id>
    <remote_id type="integer">279</remote_id>
    <node_id type="integer">2</node_id>
    <updated_at type="datetime">2016-04-28T09:38:03-07:00</updated_at>
    <readable_type>Node</readable_type>
    <ignore type="boolean">false</ignore>
  </message>
  <message>
    <id type="integer">7</id>
    <type>AuditMessage</type>
    <code type="integer">31</code>
    <message>Node live_1 is deactivated</message>
    <data nil="true"/>
    <notes nil="true"/>
    <messageable_type>Elemental::Live247::Node</messageable_type>
    <messageable_id type="integer">2</messageable_id>
    <remote_id type="integer">278</remote_id>
    <node_id type="integer">2</node_id>
```

```
<updated_at type="datetime">2016-04-28T09:17:13-07:00</updated_at>
<readable_type>Node</readable_type>
<ignore type="boolean">>false</ignore>
</message>
<message>
  <id type="integer">8</id>
  <type>AuditMessage</type>
  <code type="integer">30</code>
  <message>Node live_1 activated</message>
  <data nil="true"/>
  <notes nil="true"/>
  <messageable_type>Elemental::Live247::Node</messageable_type>
  <messageable_id type="integer">2</messageable_id>
  <remote_id type="integer">86</remote_id>
  <node_id type="integer">2</node_id>
  <updated_at type="datetime">2016-02-15T15:52:53-08:00</updated_at>
  <readable_type>Node</readable_type>
  <ignore type="boolean">>false</ignore>
</message>
</messages>
```

Example 2

The following example requests all *active code 30* (node activated) messages from *node 13*, limiting the responses to *20 per page*.

```
GET http://10.4.138.230/messages?status=active&code=30&node=13&per_page=20
```

GET System Information: Get a List of System Details

Get a list of details about the Conductor Live system, including memory, CPU, and network information.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/system_info
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response is XML content consisting of one **hash** element with the following.

- An HREF attribute that specifies the product and version installed on the Conductor Live node.
- One hash element containing several elements from the table below.

Element	Value	Description
serial-number	String	If available, provides the serial number for the hardware.
cpu-info	Array	Provides the model name and count of the CPU.
cpu-summary	String	Provides a summary of the CPU, including model number and version.
mem-info	String	Provides memory information, including: • Total memory • Used memory • Free memory • Shared memory • Buffers • Cached
network-info	Array	Provides network information such as the Ethernet ports in use.

Element	Value	Description
md-raid	String	Provides Redundant Array of Independent Disks (RAID) information, including: • RAID-Level • X • RAID-Devices • Total-Devices • State • Active-Devices • Working-Devices
hardware-raid	Array	Provides appliance hardware RAID information.
mount-info	Array	Provides information about the devices that are mounted to the AWS Elemental Conductor Live 3 system. Includes: • Device name • Path • Size • Used space • Available space • Percent space used

Example

```
GET http://198.51.100.0/system_info
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
```

```

-----
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<hash>
  <serial-number>None</serial-number>
  <cpu-info type="array">
    <cpu-info>
      <model>Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz</model>
      <count type="integer">8</count>
    </cpu-info>
  </cpu-info>
  <cpu-summary>8x Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz</cpu-summary>
  <mem-info>
    <total type="integer">8254267392</total>
    <used type="integer">5460201472</used>
    <free type="integer">2794065920</free>
    <shared type="integer">0</shared>
    <buffers type="integer">457605120</buffers>
    <cached type="integer">2555064320</cached>
  </mem-info>
  <network-info type="array">
    <network-info>02:00.0 Ethernet controller: Intel Corporation 82545EM Gigabit Ethernet Controller (Copper) (rev 01)</network-info>
    <network-info>02:01.0 Ethernet controller: Intel Corporation 82545EM Gigabit Ethernet Controller (Copper) (rev 01)</network-info>
    <network-info>02:02.0 Ethernet controller: Intel Corporation 82545EM Gigabit Ethernet Controller (Copper) (rev 01)</network-info>
  </network-info>
  <md-raid>
  </md-raid>
  <hardware-raid type="array"/>
  <mount-info type="array">
    <mount-info>
      <device>sda1</device>
      <path>/boot</path>
      <size type="integer">101529600</size>
      <used type="integer">35848192</used>
      <avail type="integer">60438528</avail>
      <percent type="integer">38</percent>
    </mount-info>
    <mount-info>
      <device>VolGroup00-LogVol00</device>
      <path>/</path>
      <size type="integer">20609396736</size>
    </mount-info>
  </mount-info>

```

```
<used type="integer">3405201408</used>
<avail type="integer">16157298688</avail>
<percent type="integer">18</percent>
</mount-info>
<mount-info>
  <device>VolGroup00-LogVol02</device>
  <path>/opt</path>
  <size type="integer">5284429824</size>
  <used type="integer">1384677376</used>
  <avail type="integer">3631316992</avail>
  <percent type="integer">28</percent>
</mount-info>
<mount-info>
  <device>VolGroup00-LogVol03</device>
  <path>/var/lib/mysql</path>
  <size type="integer">10568916992</size>
  <used type="integer">179785728</used>
  <avail type="integer">9852260352</avail>
  <percent type="integer">2</percent>
</mount-info>
<mount-info>
  <device>VolGroup00-LogVol04</device>
  <path>/data</path>
  <size type="integer">62488891392</size>
  <used type="integer">7324286976</used>
  <avail type="integer">51990355968</avail>
  <percent type="integer">13</percent>
</mount-info>
<mount-info>
  <path>total</path>
  <size type="integer">99053164544</size>
  <used type="integer">12329799680</used>
  <avail type="integer">81691670528</avail>
  <percent type="integer">14</percent>
</mount-info>
</mount-info>
</hash>
```

Configuring the Cluster

The entities in this chapter are those that you typically work with only when setting up your cluster. For background information on these entities, see the [AWS Elemental Conductor Live Configuration Guide](#).

Topics

- [Setting up Nodes](#)
- [Setting Up Routers](#)
- [Setting Up Router Inputs](#)
- [Setting Up Router Outputs](#)
- [Setting Up Redundancy Groups](#)
- [Setting Up Members of a Redundancy Group](#)
- [Setting Up a Conductor Redundancy Group](#)
- [Setting Up Members of a Conductor Redundancy Group](#)
- [Backing Up the Conductor Database](#)

Setting up Nodes

Topics

- [POST: Add a Node to the Cluster](#)
- [GET List: Get a List of Nodes in the Cluster](#)
- [GET: Get the Attributes of a Node](#)
- [DELETE: Remove a Node from the Cluster](#)

POST: Add a Node to the Cluster

HTTP Request and Response

Request URL

Add a node to the cluster.

```
POST http://<Conductor IP address>/nodes
```

Request Body

The request body is XML content consisting of one `hosts` element (of type "array") that contains:

- One or more host elements, one for each node found. Each element contains:
 - One IP address.
 - A range of IP addresses (for example, 10.4.136.[90-92]).

Call Header

- Accept: Set to `application/xml`
- Content-Type: Set to `application/xml`

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#).

Response

No response is supplied for a successful node addition. Operators should perform the GET Node operation to confirm that the node has been added successfully. For assistance, see [the section called "GET: Get the Attributes of a Node"](#).

Example

Request

This request adds four nodes: 10.4.136.15, 10.4.136.90, 10.4.136.91, 10.4.136.92.

```
POST http://198.51.100.0/nodes
-----
<?xml version="1.0" encoding="UTF-8"?>
<hosts type="array">
  <host>10.4.136.15</host>
  <host>10.4.136.[90-92]</host>
</hosts>
```

GET List: Get a List of Nodes in the Cluster

Get the list of the nodes in the cluster, including the Conductor Live node and the individual AWS Elemental Live and AWS Elemental Statmux nodes.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/nodes
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response is XML content consisting of one nodes element that contains:

- An HREF attribute that specifies the product and version installed on the Conductor Live node.
- Zero or more node elements, one for each node found (Conductor Live, AWS Elemental Live, and AWS Elemental Statmux). Each node element contains several elements.

Element	Value	Description
id	Integer	The unique ID of the node, assigned by the system when the node is created.
hostname	String	The host name of the node, assigned via a network command during setup of this appliance. Note that you do <i>not</i> assign this name

Element	Value	Description
		when adding the node to the cluster. For example, "ecle-12345678" is a typical name for an Conductor Live node. "ela-12345678" is a typical name for an AWS Elemental Live node.
ip_addr	String	The IP address of the node on the network.
status	String	The current status of the node: online, offline, failed, unknown.
product_name	String	The product name installed on this node. Always specifies: • "Conductor Live" for the Conductor Live node. • "Live" for an AWS Elemental Live node. • "Statmux" for an AWS Elemental Statmux node.
version	String	The version of the product name installed on this node.
channels	Integer	The number of channels that are associated with this node. Always "0" for the Conductor Live node.

Element	Value	Description
inflight_channels	Integer	Number of channels currently in use for encoding. Always 0 for a Conductor node, 0 or a number for a worker node.
mptses	Integer	Number of MPTS outputs associated with this node. Always 0 for a Conductor node, 0 or a number for a worker node.
active_alerts	Integer	Number of alerts that are associated with that node and that are active (they have not been automatically cleared by the system).
recent_error_messages	Integer	Number of error messages that are associated with that node and that have occurred in the last 168 hours (one week). Does not include content messages, audit messages or warning messages.

Element	Value	Description
redundancy_group		The ID assigned to the redundancy group, the name of the redundancy group, and the name of the product running on the worker nodes (values are “live” and “statmux”). If this node is a worker node and belongs to a redundancy group, contains the elements for that group. See the section called “GET List: Get a List of Conductor Redundancy Group Members”. If this node is a Conductor node, nil.
authentication	Integer	If user authentication is set up on this node, specifies the ID of the “API user” that you created when you set up for authentication, as described in the AWS Elemental Conductor Live Configuration Guide. In effect, a value here tells you that user authentication is enabled on this node.

Example

The response to this request specifies that there are three nodes set up in the cluster: one Conductor Live node (as indicated by the `product_name`) and two AWS Elemental Live nodes. All the nodes have user authentication enabled. In this example, the Conductor Live has the ID 1, but you cannot rely on this node always being assigned that ID.

```
GET http://198.51.100.0/nodes
```

```
-----  
Content-type:application/vnd.elemental+xml;version=3.3.0  
-----
```

```
<?xml version="1.0" encoding="UTF-8"?>  
<nodes href="/nodes" product="AWS Elemental Conductor Live" version="3.3.nnnnn">  
  <node>  
    <id>1</id>  
    <hostname>ecle-12345678</hostname>  
    <ip_addr>198.51.100.0</ip_addr>  
    <status>active</status>  
    <product_name>AWS Elemental Conductor Live</product_name>  
    <version>3.3.nnnnn</version>  
    <channels>0</channels>  
    <inflight_channels type="integer">0</inflight_channels>  
    <mptses type="integer">0</mptses>  
    <active_alerts type="integer">0</active_alerts>  
    <recent_error_messages type="integer">0</recent_error_messages>  
    <redundancy_group nil="true"/>  
    <authentication>  
      <user_id type="integer">1</user_id>  
    </authentication>  
  </node>  
  <node>  
    <id>2</id>  
    <hostname>ecle-12345678</hostname>  
    <ip_addr>10.4.136.91</ip_addr>  
    <status>active</status>  
    <product_name>Live</product_name>  
    <version>2.7.0.67890</version>  
    <channels>4</channels>  
    <inflight_channels type="integer">1</inflight_channels>  
    <mptses type="integer">3</mptses>  
    <active_alerts type="integer">2</active_alerts>  
    <recent_error_messages type="integer">0</recent_error_messages>  
    <redundancy_group nil="true"/>  
  </node>  
</nodes>
```

```
<authentication>
  <user_id type="integer">1</user_id>
</authentication>
</node>
<node>
  <id>3</id>
  <hostname>ecle-22334455</hostname>
  <ip_addr>10.4.136.92</ip_addr>
  <status>active</status>
  <product_name>Live</product_name>
  <version>2.7.0.67890</version>
  <channels>3</channels>
  <inflight_channels type="integer">0</inflight_channels>
  <mptses type="integer">0</mptses>
  <active_alerts type="integer">0</active_alerts>
  <recent_error_messages type="integer">0</recent_error_messages>
  <redundancy_group nil="true"/>
  <authentication>
    <user_id type="integer">1</user_id>
  </authentication>
</node>
</nodes>
```

GET: Get the Attributes of a Node

Get the attributes on the specified node.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/nodes/<ID of node>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#) .

Response

The response is XML content consisting of one node element, containing the same elements as the response for GET Node List above.

Example

This request gets the attributes for the node that has the ID 3.

```
GET http://198.51.100.0/nodes/3
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
-----
<?xml version="1.0" encoding="UTF-8"?>
<node>
  <id>3</id>
  <hostname>ecle-22334455</hostname>
  <ip_addr>10.4.136.92</ip_addr>
  <status>active</status>
  <product_name>Live</product_name>
  <version>2.7.0.67890</version>
  <channels>3</channels>
  <inflight_channels type="integer">0</inflight_channels>
  <mptses type="integer">0</mptses>
  <active_alerts type="integer">0</active_alerts>
  <recent_error_messages type="integer">0</recent_error_messages>
  <redundancy_group nil="true"/>
  <authentication>
    <user_id type="integer">1</user_id>
  </authentication>
</node>
```

DELETE: Remove a Node from the Cluster

Remove the specified node from the Conductor Live cluster. The node cannot be associated with any channels, so follow this procedure.

1. Send a GET Channel List request to see which channels use this node.
2. Send a POST Stop request to stop the channels that are running.
3. Send a PUT Channel request on each channel to modify it so that it does not use this node.
4. Delete the channel as described in [the section called "DELETE: Delete a Channel"](#).

HTTP Request and Response

Request URL

```
DELETE http://<Conductor IP address>/nodes/<ID of node>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Example

This request removes the node with the ID 8.

```
DELETE http://198.51.100.0/nodes/8
```

Setting Up Routers

The Router element holds information about a router that is attached to SDI cards on the AWS Elemental Live node.

Topics

- [POST: Create a Router](#)
- [PUT: Modify a Router](#)
- [GET List: Get a List of Routers](#)
- [GET: Get Router Attributes](#)
- [DELETE: Delete a Router](#)

POST: Create a Router

This request creates a new router to correspond to a router that is connected to HD-SDI inputs on one or more AWS Elemental Live nodes. (The same router can have connections to several nodes.)

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/routers
```

Call Header

- Accept: Set to `application/xml`
- Content-Type: Set to `application/xml`

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The request body is XML content consisting of one `router` element, consisting of the following elements.

Element	Value	Description
name	String	This name appears in the Inputs drop-down list.
ip	String	The IP address without any protocol.
max_inputs	Integer	The number of inputs you want to use on this router. This number must be less than or equal to the number of physical inputs on the router.
max_outputs	Integer	The number of outputs you want to use on this router. This number must be less than or equal to the number

Element	Value	Description
		of physical outputs on the router.
router_type	String	One of these: • blackmagic_videohub • miranda_nvision • harris_panacea • snell_aurora
level_id	Integer	This appears only for Harris Panacea and Miranda nVision.
user_id	Integer	This appears only for Miranda nVision.
matrix_id	Integer	This appears only for Snell Aurora.

Response

The response repeats back the data that you posted with the addition of <id>: the newly assigned ID for the router.

The response is identical to the response to a GET Router. See the following example.

Example

Request

This request creates one router with the name "SDI_Router." This router has 12 inputs and 12 outputs, so these numbers are set in the max_inputs and max_outputs.

```
POST http://198.51.100.0/routers
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
```

```
<router>
  <name>SDI_Router</name>
  <ip>192.168.10.10</ip>
  <router_type>blackmagic_videohub</router_type>
  <max_inputs>12</max_inputs>
  <max_outputs>12</max_outputs>
</router>
```

Response

In this example, the router is given the ID 1.

```
<?xml version="1.0" encoding="UTF-8"?>
<router href="/routers" product="AWS Elemental Conductor Live" version="3.3.nnnnn">
  <id>1</id>
  <name>BlackMagic1</name>
  <ip>192.168.10.10</ip>
  <router_type>blackmagic_videohub</router_type>
  <max_inputs>12</max_inputs>
  <max_outputs>12</max_outputs>
</router>
```

PUT: Modify a Router

Modify the attributes of the specified router. You cannot use this command to modify the inputs or outputs of the router; to do that, use PUT Router Input and PUT Router Output.

HTTP Request and Response

Request URL

```
PUT http://<Conductor IP address>/routers/<ID of router>
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#).

Request Body

The request body contains only the elements to change. For a list of available elements, see [the section called "POST: Create a Router"](#).

Example

This request changes the router with the ID 3. It changes its max_inputs to 8.

```
PUT http://198.51.100.0/routers/3
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<router>
  <router_type>blackmagic_videohub</router_type>
  <max_inputs>8</max_inputs>
</router>
```

GET List: Get a List of Routers

Get a list of all video SDI routers, including the data that is contained in the Router Input and Router Output entities.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/routers
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#).

Response

The response is XML content consisting of one routers element that contains:

- An HREF attribute that specifies the product and version installed on the Conductor Live node.
- Zero or more `router` elements, one for each router found. Each router element consists of several elements.

Element	Value	Description
id	Integer	The ID for this router, assigned by the system when the router is created.
name	See the section called “POST: Create a Router” .	
ip		
router_type		
level_id		
user_id		See the section called “POST: Create a Router Input” .
id		
name		
router_id		
input_number		
id		See the section called “POST: Create a Router Output” .
output_number		
router_id		
device_id		

Example

Request

```
GET http://198.51.100.0/routers
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<routers href="/routers" product="AWS Elemental Conductor Live" version="3.3.nnnnn">>
  <router>
    <id>1</id>
    <name>BlackMagic1</name>
    <ip>192.168.10.10</ip>
    <router_type>blackmagic_videohub</router_type>
    <max_inputs>12</max_inputs>
    <max_outputs>12</max_outputs>
    <inputs>
      <input>
        <id>1</id>
        <name>Input 1</name>
        <router_id>1</router_id>
        <input_number>1</input_number>
      </input>
      <input>
        <id>2</id>
        <name>Input 2</name>
        <router_id>1</router_id>
        <input_number>2</input_number>
      </input>
    </inputs>
    <outputs>
      <output>
        <id>9</id>
        <router_id>1</router_id>
        <output_number>1</output_number>
        <device_id>1</device_id>
        <device_type>Device</device_type>
      </output>
    </outputs>
  </router>
  .
  .
```

```
.  
  <router>  
.    
.    
.    
  </router>  
</routers>
```

GET: Get Router Attributes

Get the attributes of the specified video SDI router, including the data that is contained in the Router Input and Router Output.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/routers/<ID of router>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response contains XML content consisting of one router element, containing the same elements as the response for [the section called “GET List: Get a List of Routers”](#).

Example

This response shows the information for the router with the ID 1. For this router, two inputs have been set up, but, so far, no outputs have been set up. The outputs would have to be set up in order for the router to be useable as an input source.

```
GET http://198.51.100.0/routers/1  
-----  
Content-type:application/vnd.elemental+xml;version=3.3.0
```

```
-----
<?xml version="1.0" encoding="UTF-8"?>
<router href="/routers/2" product="AWS Elemental Conductor Live" version="3.3.nnnnn">
  <id>2</id>
  <name>SDI_Router</name>
  <ip>192.168.10.12</ip>
  <router_type>harris_panacea</router_type>
  <max_inputs>12</max_inputs>
  <max_outputs>12</max_outputs>
  <level_id>0</level_id>
  <inputs>
    <input>
      <id>3</id>
      <name>Input 1</name>
      <router_id>2</router_id>
      <input_number>1</input_number>
    </input>
    <input>
      <id>4</id>
      <name>Input 2</name>
      <router_id>2</router_id>
      <input_number>2</input_number>
    </input>
  </inputs>
  <outputs/>
</router>
```

DELETE: Delete a Router

Deletes the specified router (identified by its internal router ID) and the associated inputs and outputs. To get the internal router ID of a specific router, see [the section called "GET: Get Router Attributes"](#).

HTTP Request and Response

Request URL

```
DELETE http://<Conductor IP address>/routers/<ID of router>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Example

This request deletes the router with the ID 2.

```
DELETE http://198.51.100.0/routers/2
```

Setting Up Router Inputs

The Router Inputs entity holds information about the router inputs you are using with an AWS Elemental Live node. You only need to add information about the router inputs that are being or will be used (that are hooked up to an AWS Elemental Live node).

Topics

- [POST: Create a Router Input](#)
- [PUT: Modify a Router Input](#)
- [GET List: Get a List of Router Inputs](#)
- [GET: Get Attributes of a Router Input](#)
- [DELETE: Delete a Router Input](#)

POST: Create a Router Input

This request creates a new input for the specified router. Use this command repeatedly to set up all the router inputs. The total number of Router Input entities for a specified router must not exceed the `max_input` on that Router entity: it must not exceed the actual physical inputs on the router.

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/routers/<ID of router>/inputs
```

Call Header

- Accept: Set to `application/xml`

- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The response is in XML content consisting of one input element, consisting of the following elements.

Element	Value	Description
name	String	A name for this input, of your choosing. This name appears in the Inputs drop-down list on the Conductor Live web interface.
input_number	Integer	The ID of the router input that you want to make known to Conductor Live. This is the ID for the input as assigned by the router (not assigned by AWS Elemental software). When adding 4 Quadrant-4k inputs for HEVC, enter the first input number in the sequence.
input_number_end	Integer	When adding 4 Quadrant-4k inputs for HEVC, enter the last (fourth) input number in the sequence. For example, if you provided “1” for input_number, you

Element	Value	Description
		would use "4" for input_number_end. This element is not used if you are not adding 4 Quadrant-4k inputs.
quad	True/False	It is not necessary to specify this if you are not adding 4 Quadrant-4k inputs.

Response

The response repeats back the data that you posted with the addition of:

- `id`: The newly assigned ID for the router_input.
- `router_id`: The router that this router input belongs to.

The response is identical to the response to a GET Router Input. See below for an example.

Example

Request

This request is to create a router input for the input that has the router-assigned ID of 1.

```
POST http://198.51.100.0/routers/2/inputs
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
<input>
  <name>Input 1</name>
  <input_number>1</input_number>
</input>
```

Response

In this example, the router input has been automatically assigned the ID 4. The already existing ID of the router itself is 2.

```
<input>
  <id>4</id>
  <name>Input 1</name>
  <router_id>2</router_id>
  <input_number>1</input_number>
  <input_number_end nil="true"/>
  <quad>false</quad>
</input>
```

PUT: Modify a Router Input

Modify the attributes of the specified input on the specified router. If, after the initial setup, you ever change the cabling on the input side of your router, you must use PUT Router Input to reflect these changes.

HTTP Request and Response

Request URL

```
PUT http://<Conductor IP address>/routers/<ID of router>/inputs/<ID of input>
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#).

Request Body

The body contains only the elements to change. For a list of all elements, see [the section called "POST: Create a Router"](#)

Example

This request changes the router input number (the number of the physical input port on the router) to 4. The number 3 at the end of the example URL is the id assigned by Conductor Live when this input was created in the software. This input belongs to the router with the ID of 2.

This change would only be made to fix an error in the original setup or to reflect a change in the cabling (so that the router's 4th input is now being used).

```
PUT http://198.51.100.0/routers/2/inputs/3
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<input>
  <input_id>4</input_id>
</input>
```

GET List: Get a List of Router Inputs

Get the list of all the inputs for the specified router.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/routers/<ID of router>/inputs
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#).

Response

The response is XML content consisting of one inputs element that contains:

- An HREF attribute that specifies the product and version installed on the Conductor Live node.
- Zero or more `input` elements, one for each input found. Each `input` element contains several elements.

Element	Value	Description
<code>id</code>	Integer	The ID for this input, unique for this router and assigned by AWS Elemental Conductor Live 3 when the input is created.
<code>name</code>	String	The name for this input.
<code>router_id</code>	Integer	The router to which this input belongs.
<code>input_number</code>	Integer	The ID of the router input corresponding to the physical port on the router. This number is assigned by the router and specified to Conductor Live by the user (not assigned by AWS Elemental software). For 4 Quadrant-4k inputs, this is the first input number in a sequence of four.
<code>input_number_end</code>	Integer	For 4 Quadrant-4k inputs, the last input number in the sequence of four. This element is not used if you are not adding 4 Quadrant-4k inputs.

Element	Value	Description
quad	True/False	It is not necessary to specify if you are not adding 4 Quadrant-4k inputs.

Example

This request is for the inputs associated with the router with the ID 4. The response specifies that there are two inputs in this router with IDs 3 and 4.

```
GET http://198.51.100.0/routers/4/inputs
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
-----
<?xml version="1.0" encoding="UTF-8"?>
<inputs href="/routers/4/inputs" product="AWS Elemental Conductor Live"
version="3.3.nnnnn">
  <input>
    <id>3</id>
    <name>Input 1</name>
    <router_id>2</router_id>
    <input_number>1</input_number>
    <input_number_end nil="true"/>
    <quad>false</quad>
  </input>
  <input>
    <id>4</id>
    <name>Input 2</name>
    <router_id>2</router_id>
    <input_number>2</input_number>
    <input_number_end nil="true"/>
    <quad>false</quad>
  </input>
</inputs>
```

GET: Get Attributes of a Router Input

Get the attributes of the specified input on the specified router.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/routers/<ID of router>/inputs/<ID of input>
```

Call Header

- Accept: Set to application/xml.

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response is XML content consisting of one input element, with the same elements as the response for [the section called “GET: Get Router Attributes”](#).

Example

This request gets the attributes for the input that has the ID 3. The input belongs to the router with the ID 4.

```
GET http://198.51.100.0/routers/4/inputs/3
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
-----
<?xml version="1.0" encoding="UTF-8"?>
<input href="/routers/4/inputs/3" product="AWS Elemental Conductor Live"
version="3.3.nnnnn">
  <id>3</id>
  <name>Input 1</name>
  <router_id>4</router_id>
  <input_number>1</input_number>
  <input_number_end nil="true"/>
  <quad>false</quad>
</input>
```

DELETE: Delete a Router Input

Delete the specified input on the specified router. To get the ID of the input, use GET Router List and look for the <id> element (not the <input_number> element!).

HTTP Request and Response

Request URL

```
DELETE http://<Conductor IP address>/routers/<ID of router>/inputs/<ID of input>
```

Call Header

- Accept: Set to application/xml.

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Example

This request deletes the input with the ID 1 from the router with the ID 2.

```
DELETE http://198.51.100.0/routers/2/inputs/1
```

Setting Up Router Outputs

The Router Outputs entity holds information about each router output that is connected to an SDI card on an AWS Elemental Live appliance. The information is a mapping from the router output to the SDI input.

If, after the initial setup, you ever hook up another cable between a router output and an SDI input, you must set it up using POST Router Input.

Topics

- [POST: Create a Router Output](#)
- [PUT: Modify a Router Output](#)
- [GET List: Get Router Output List](#)
- [GET: Get Attributes of a Router Output](#)

- [DELETE: Delete a Router Output](#)

POST: Create a Router Output

Create a new output for the specified router. Use this command repeatedly to set up all the router outputs. The total number of Router Output entities for a specified router must not exceed the actual physical outputs on the router.

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/routers/<ID of router>/outputs
```

Call Header

- Accept: Set to `application/xml`
- Content-Type: Set to `application/xml`

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The request body contains XML content consisting of one output element, consisting of the following elements.

Element	Value	Description
output_number	Integer	An ID on the router.
		The ID of the router output that is connected to an SDI card on the AWS Elemental Live node. You specify the router by its output ID as assigned by the router (not assigned by AWS Elemental

Element	Value	Description
		software). Check the manufacturer’s documentation for the numbering scheme for the router. For example, the manufacturer may number ports starting from 1 and starting from the upper left. In this case, the second output from the left might be numbered “2.” When adding 4 Quadrant-4k outputs for HEVC, enter the first output number in the sequence.
output_number_end	Integer	When adding 4 Quadrant-4k outputs for HEVC, enter the last (fourth) output number in the sequence. For example, if you entered “1” for output_number, you would use “4” for output_number_end. This element is not used if you are not adding the 4 Quadrant-4k outputs.

Element	Value	Description
device_id	String	An ID on the SDI card. The ID of the SDI input that is connected to the router output identified by output_number. This ID is assigned by AWS Elemental Live and provided to Conductor Live when the SDI card is auto-detected. Each port on the entire node is assigned a unique ID (for example, IDs 1-5 on the first card, 6-10 on the second). To obtain the device ID, see the Settings > Input Devices page of the Conductor web interface. The Conductor API does not expose this information.

Response

The response repeats back the data that you posted with the addition of:

- id: The newly assigned ID for the router.
- router_id: The router to which this router output belongs.

The response is identical to the response to a GET Router Output. See below for an example.

Example

Request

This request creates one output for the router with the ID 2. It specifies that router output ID 3 is connected to SDI card input ID 4.

```
POST http://198.51.100.0/routers/2/outputs
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
<output>
  <output_number>3</output_number>
  <device_id>4</device_id>
</output>
```

Response

In this example, the router output is given the ID 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<output href="/routers/2/output" product="AWS Elemental Conductor Live"
version="3.3.nnnnn">
  <id>4</id>
  <router_id>1</router_id>
  <output_number>2</output_number>
  <output_number_end nil="true"/>
  <device_id>5</device_id>
  <device_type>Device</device_type>
</output>
```

PUT: Modify a Router Output

Modify the attributes of the specified output on the specified router. Modify the attributes if a cable from the router output moves to a different SDI card input.

HTTP Request and Response

Request URL

```
PUT http://<Conductor IP address>/routers/<ID of router>/outputs/<ID of output>
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#).

Request Body

The body contains only the elements to change. For available elements, see [the section called "POST: Create a Router"](#).

Example

This request changes the router output with the ID 4. This input belongs to the router with the ID 2. It changes its device_id to 3 to indicate that the connection represented by this router output is actually to the SDI input that has the ID 3.

```
PUT http://198.51.100.0/routers/2/outputs/4
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<output>
  <device_ID>3</device_id>
</output>
```

GET List: Get Router Output List

Get the list of outputs for the specified router.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/routers/<ID of router>/outputs
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response contains XML content consisting of one outputs element with the following.

- An HREF attribute that specifies the product and version installed on the Conductor Live node.
- Zero or more output elements, one for each input found. Each output element contains several elements.

Element	Value	Description
id	Integer	The ID for this output, unique for this router and assigned by the system when the output is created.
output_number	Integer	The ID of the router output that is connected to the SDI card identified by device_id (below). This ID is the ID assigned by the router (not as assigned by AWS Elemental Live).
router_id	Integer	The router that this output belongs to-- a unique ID assigned by AWS Elemental Live.
device_id	String	The ID the SDI input that is connected to the router output identified by

Element	Value	Description
		output_number. This ID is assigned by AWS Elemental Live when the SDI card is auto-detected. Each port on the entire node is assigned a unique ID (for example, IDs 1-4 on the first card, 5-8 on the second).
device_type	String	A string which always specifies "Device."

Example

This request is for the outputs associated with the router with the ID 2. The response specifies that there are two outputs in this router with IDs 3 and 4.

```
GET http://198.51.100.0/routers/2/outputs
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
-----
<?xml version="1.0" encoding="UTF-8"?>
<outputs href="/routers/2/outputs" product="AWS Elemental Conductor Live"
version="3.3.nnnnn">
  <output>
    <id>3</id>
    <output_number>5</output_number>
    <router_id>2</router_id>
    <device_id>1</device_id>
  </output>
  <output>
    <id>4</id>
    <output_number>2</output_number>
    <router_id>2</router_id>
    <device_id>7</device_id>
  </output>
</outputs>
```

GET: Get Attributes of a Router Output

Get the attributes of the specified output on the specified router.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/routers/<ID of router>/outputs/<ID of output>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#).

Response

The response is XML content consisting of one output element, containing the same elements as the response for [the section called "GET: Get Router Attributes"](#).

Example

This request gets the attributes for the output that has the ID 3. The input belongs to the router with this ID 4.

```
GET http://198.51.100.0/routers/4/outputs/3
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
-----
<?xml version="1.0" encoding="UTF-8"?>
<output href="/routers/4/outputs/3" product="AWS Elemental Conductor Live"
  version="3.3.nnnnn">
  <id>3</id>
  <router_id>4</router_id>
  <output_number>1</output_number>
  <device_id>1</device_id>
  <device_type>Device</device_type>
</output>
```

DELETE: Delete a Router Output

Delete the specified output on the specified router.

HTTP Request and Response

Request URL

```
DELETE http://<Conductor IP address>/routers/<ID of router>/outputs/<ID of output>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Example

This request deletes the output with the ID 1 from the router with the ID 2.

```
DELETE http://198.51.100.0/routers/2/outputs/1
```

Setting Up Redundancy Groups

Topics

- [POST: Create a Redundancy Group](#)
- [PUT: Modify a Redundancy Group](#)
- [GET List: Get a List of Redundancy Groups](#)
- [GET: Get Attributes of a Redundancy Group](#)
- [DELETE: Delete a Redundancy Group](#)

POST: Create a Redundancy Group

Create a new redundancy group for AWS Elemental Live or AWS Elemental Statmux redundancy group with the specified attributes.

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/redundancy_groups
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The request contains XML content consisting of one `redundancy_group` element with the following elements.

Element	Value	Description
name	String	A name you assign. This must be unique in the cluster.
product_name	String	A name which must specify “live” if nodes in this group are running AWS Elemental Live; specify “statmux” for nodes running AWS Elemental Statmux. This value is case-sensitive and requires all letters to be in lower case. All the nodes that you add to the redundancy group must have the same product_name.

Response

The response repeats back the data that you posted with the addition of:

- id: The newly assigned ID for the redundancy group.

The response is identical to the response to a GET Redundancy Group. For an example, see [the section called “GET List: Get a List of Redundancy Groups”](#).

Example

Request

This request creates one redundancy group named “backup” with its product set as “live” and its product version set at “2.7.0.12345.”

```
POST http://198.51.100.0/redundancy_groups
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<redundancy_group>
  <name>backup</name>
  <product_name>live</product_name>
</redundancy_group>
```

PUT: Modify a Redundancy Group

Change the name of the specified redundancy group.

HTTP Request and Response

Request URL

```
PUT http://<Conductor IP address>/redundancy_groups/<ID of redundancy group>
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The body contains only the elements to change. For the format, see [the section called “POST: Create a Redundancy Group”](#).

Example

This request changes the name of the redundancy group with the ID 1. It changes the name to “backup_nodes.”

```
PUT http://198.51.100.0/redundancy_groups/1
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<redundancy_group>
<name>backup_nodes</name>
</redundancy_group>
```

GET List: Get a List of Redundancy Groups

Get a list of all redundancy groups, including the attributes of each group.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/redundancy_groups
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response is XML content consisting of one `redundancy_groups` element with the following .

- An HREF attribute that specifies the product and version installed on the Conductor Live node.
- Zero or more `redundancy_group` elements, one for each group found. Each `redundancy_group` element contains several elements.

Element	Value	Description
id	Integer	The ID of this redundancy group.
name	String	The name you assigned to the group.
product_name	String	The software type running on the nodes of the redundancy group. Values are "live" or "statmux."

Example

The response to this request shows two redundancy groups named "2.5 nodes" (intended for AWS Elemental Live nodes running version 2.5.x) and "2.4 nodes" (intended for AWS Elemental Live nodes running version 2.4.x).

```
GET http://198.51.100.0/redundancy_groups
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
-----
<?xml version="1.0" encoding="UTF-8"?>
<redundancy_groups href="/redundancy_groups" product="AWS Elemental Conductor Live"
version="3.3.43776">
  <redundancy_group>
    <id>1</id>
    <name>2.5 nodes</name>
    <product_name>live</product_name>
```

```
</redundancy_group>
  <id>3</id>
  <name>2.4 nodes</name>
  <product_name>live</product_name>
</redundancy_group>
</redundancy_groups>
```

GET: Get Attributes of a Redundancy Group

Get the attributes of the specified redundancy group.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/redundancy_groups/<ID of redundancy group>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#).

Response

The response is XML content consisting of one redundancy_group element, with the same elements as the response for [the section called "GET List: Get a List of Redundancy Groups"](#).

Example

This response shows the attributes for the group that has the ID 1.

```
GET http://198.51.100.0/redundancy_groups/1
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
-----
<?xml version="1.0" encoding="UTF-8"?>
<redundancy_group>
  <id>1</id>
```

```
<name>2.5 nodes</name>
<product_name>live</product_name>
</redundancy_group>
```

DELETE: Delete a Redundancy Group

Delete the redundancy group that has the specified ID.

HTTP Request and Response

Request URL

```
DELETE http://<Conductor IP address>/redundancy_groups/<ID of redundancy group>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Example

This request deletes the redundancy group with the ID 2 .

```
DELETE http://198.51.100.0/redundancy_groups/2
```

Setting Up Members of a Redundancy Group

Topics

- [POST: Add a Node to a Redundancy Group](#)
- [PUT: Change Role of a Member of a Redundancy Group](#)
- [GET List: Get a List of Redundancy Group Members](#)
- [GET: Get the Attributes of a Redundancy Group Member](#)
- [DELETE: Remove a Node from a Redundancy Group](#)
- [POST Initiate Failover](#)

POST: Add a Node to a Redundancy Group

Add a node to the specified redundancy group and assign the node’s role as active or backup. The redundancy group must already exist.

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/redundancy_groups/<ID of redundancy group>/members
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The request contains XML content consisting of one redundancy_groups element with the following.

- One or more redundancy_group_member elements, each containing the following elements.

Element	Value	Description
node_id	Integer	The ID of the node to add. Use GET Node List to get a list that shows the ID of each node. A node can belong to only one redundancy group. If the node’s role will be “backup,” the node must have

Element	Value	Description
		no channels associated with it. (You get a "Node is not reservable" error.) To verify its channels, do a GET Channel List and verify if this node is listed as associated with a channel; if it is, do a PUT Channel on that channel so that the channel uses a different node.
role	String	The initial role of the node in the redundancy group: "active" or "backup." • A backup node automatically switches to "active" if it is selected as a failover node. It then remains active until you change its role (perhaps by doing PUT Member). • An active node never automatically changes its role.

Response

The response repeats back the data that you posted with the addition of:

- id: The newly assigned ID for the member.

The response is identical to the response to a GET Redundancy Member. See below for an example.

Example

Request

Add the node with the ID 2 to the redundancy group that has the ID 3.

```
POST http://198.51.100.0/redundancy_groups/3/members
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<redundancy_group_member>
<node_id>2</node_id>
<role>active</role>
<redundancy_group_position>1</redundancy_group_position>
</redundancy_group_member>
```

Response

The response shows the “redundancy group member” data and the standard node data that was applied to this node. Note that the node has the member ID of “3” (its ID in the redundancy group) but has the node ID of “2” (its ID in the cluster). For details, see [the section called “GET List: Get a List of Redundancy Group Members”](#).

```
<?xml version="1.0" encoding="UTF-8"?>
<redundancy_group_member href="/redundancy_groups/3/members" product="AWS Elemental
Conductor Live" version="3.3.nnnnn">
  <id type="integer">3</id>
  <role>active</role>
  <node>
    <id type="integer">2</id>
    <hostname>live_3</hostname>
    <ip_addr>10.4.138.233</ip_addr>
    <product_name>Live</product_name>
    <status>online</status>
    <version>2.9.2.40404</version>
    <channels type="integer">3</channels>
    <inflight_channels type="integer">0</inflight_channels>
    <mptses type="integer">1</mptses>
    <active_alerts type="integer">0</active_alerts>
    <recent_error_messages type="integer">0</recent_error_messages>
```

```
</node>  
</redundancy_group_member>
```

PUT: Change Role of a Member of a Redundancy Group

Modify the role of the specified node in the specified redundancy group.

HTTP Request and Response

Request URL

```
PUT http://<Conductor IP address>/redundancy_groups/<ID of redundancy group>/members/  
<ID of member node>
```

Note

Identify the member to change by specifying the redundancy group member ID within the redundancy group, not the ID within the cluster (the node ID). See the table under [the section called “GET List: Get a List of Redundancy Group Members”](#) for an explanation of how to get each kind of ID.

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The request contains XML content consisting of:

- One redundancy_group_member with
- role: “active” or “backup”

Example

This request changes a property of the redundancy group member identified by the member ID 3 that is in the redundancy group with the ID 1. It changes the node's role from "backup" to "active."

```
PUT http://198.51.100.0/redundancy_groups/1/members/3
```

```
-----  
<?xml version="1.0" encoding="UTF-8"?>  
<redundancy_group_member>  
<role>active</role>  
</redundancy_group_member>
```

GET List: Get a List of Redundancy Group Members

Get the list of the nodes that are members of the specified redundancy group.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/redundancy_groups/<ID of redundancy group>/members
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#).

Response

The response is XML content consisting of one `redundancy_group_members` element with the following.

- An HREF attribute that specifies the product and version installed on the Conductor Live node.
- Zero or more `redundancy_group_member` elements, one for each group found. Each `redundancy_group_member` element contains several elements:

Element	Value	Description
id	Integer	The member ID for this node.  Note Within <redundancy_group_members>, <id> is the ID for this node as identified inside the redundancy group, assigned with the node that was added to the redundancy group. Within <node>, <id> is the unique ID for this node as identified in all of Conductor Live, assigned when the node was added to the cluster.
role	String	The current role of the node in the redundancy group: “active” or “backup.”
node	Object	The node information for this member of the redundancy group; see the section called “Setting up Nodes” .

Example

The response shows that the redundancy group with the ID 1 has two members—one backup and one active.

GET http://198.51.100.0/redundancy_groups/1/members

Content-type:application/vnd.elemental+xml;version=3.3.0

```
<?xml version="1.0" encoding="UTF-8"?>
<redundancy_group_members href="/redundancy_groups/1/members" product="AWS Elemental
Conductor Live" version="3.3.nnnnn">
  <redundancy_group_member>
    <id type="integer">5</id>
    <role>active</role>
    <node>
      <id type="integer">2</id>
      <hostname>live_1</hostname>
      <ip_addr>10.4.138.233</ip_addr>
      <product_name>Live</product_name>
      <status>online</status>
      <version>2.9.2.40404</version>
      <channels type="integer">0</channels>
      <inflight_channels type="integer">0</inflight_channels>
      <mptses type="integer">1</mptses>
      <active_alerts type="integer">0</active_alerts>
      <recent_error_messages type="integer">0</recent_error_messages>
    </node>
  </redundancy_group_member>
  <redundancy_group_member>
    <id type="integer">6</id>
    <role>backup</role>
    <node>
      <id type="integer">3</id>
      <hostname>live_2</hostname>
      <ip_addr>10.4.138.234</ip_addr>
      <product_name>Live</product_name>
      <status>online</status>
      <version>2.9.2.40404</version>
      <channels type="integer">0</channels>
      <inflight_channels type="integer">0</inflight_channels>
      <mptses type="integer">0</mptses>
      <active_alerts type="integer">0</active_alerts>
      <recent_error_messages type="integer">0</recent_error_messages>
    </node>
  </redundancy_group_member>
</redundancy_group_members>
```

GET: Get the Attributes of a Redundancy Group Member

Get the attributes of the specified member in the specified redundancy group.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/redundancy_groups/<ID of redundancy group>/members/  
<ID of member node>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response is XML content consisting of one redundancy_group element containing the same elements as the response for [the section called “GET List: Get a List of Redundancy Groups”](#).

Example

This request gets the attributes for the redundancy group member that has the ID 3 within the redundancy group and that belongs to the redundancy group with the ID 1. The node's ID independent of redundancy group is 2.

```
GET http://198.51.100.0/redundancy_groups/1/members/3  
-----  
Content-type:application/vnd.elemental+xml;version=3.3.0  
-----  
<?xml version="1.0" encoding="UTF-8"?>  
<redundancy_group_member href="/redundancy_groups/1/members/3" product="AWS Elemental  
Conductor Live" version="3.3.nnnnn">  
  <id type="integer">3</id>  
  <role>active</role>  
  <node>  
    <id type="integer">2</id>  
    <hostname>live_1</hostname>
```

```
<ip_addr>10.4.138.233</ip_addr>
<product_name>Live</product_name>
<status>online</status>
<version>2.9.2.40404</version>
<channels type="integer">0</channels>
<inflight_channels type="integer">0</inflight_channels>
<mptses type="integer">1</mptses>
<active_alerts type="integer">0</active_alerts>
<recent_error_messages type="integer">0</recent_error_messages>
</node>
</redundancy_group_member>
```

DELETE: Remove a Node from a Redundancy Group

Remove the node with the specified member ID from the specified redundancy group.

```
DELETE http://<Conductor IP address>/redundancy_groups/<ID of redundancy group>/
members/<ID of member node>
```

POST Initiate Failover

Initiate failover of the specified node. The node must be part of a redundancy group, must have an "active" role, and must be running a channel. The redundancy group must also have at least one backup node.

When failover is initiated, the node becomes "idle" and all its channels are moved to a backup node. That backup node becomes an active node.

To make the original node active again, associate channels with the node again (using the PUT Channel command) and change that node's status to "active."

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/nodes/<ID of node>/backup
```

Call Header

- Accept: Set to application/xml

- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Example

This request initiates failover the node with the ID 8.

```
POST http://198.51.100.0/nodes/8/backup
```

Setting Up a Conductor Redundancy Group

Topics

- [POST: Create a Conductor Redundancy Group](#)
- [PUT: Modify a Conductor Redundancy Group](#)
- [GET: Get the Attributes of the Conductor Redundancy Group](#)
- [DELETE: Delete a Conductor Redundancy Group](#)
- [POST Enable: Enable Conductor Redundancy Group](#)
- [DELETE Disable: Disable Conductor Redundancy Group](#)

POST: Create a Conductor Redundancy Group

Create a new Conductor redundancy group with the specified attributes. You can create only one Conductor redundancy group.

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/conductor_redundancy_groups
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The response contains XML content consisting of one `conductor_redundancy_group` element with the following elements.

Element	Value	Description
<code>vip</code>	String	A valid ipv4 address. This address is the “cluster ID” for the two Conductor nodes. This address must be an address on your network that is never allocated to any other host.
<code>vrid</code>	Integer	The virtual router ID, a number from 1-254. This ID must not conflict with any other instance of keepalived (or another (Virtual Router Redundancy Protocol)VRRP-based service) that is running on your network. To check for conflicting VRRP services, use: <pre>sudo tcpdump vrrp</pre> If you see VRRP advertisement packets, do not use the listed VRIDs.

Element	Value	Description
name	String	Choose a name.

Response

The response repeats back the data that you posted with the addition of the following.

- **id:** The newly assigned ID for the Conductor redundancy group.
- **enabled:** This setting is always "false" on a newly created group to indicate that Conductor redundancy is not yet enabled.
- **product_name:** This setting is always "conductor_live."

The response is identical to the response to a GET Conductor Redundancy Group. For an example, see [the section called "GET: Get the Attributes of the Conductor Redundancy Group"](#).

Example

Request

This request creates one Conductor redundancy group that has the VIP 10.4.200.200 and the VRID 2.

```
POST http://198.51.100.0/conductor_redundancy_groups
-----
<?xml version="1.0" encoding="UTF-8"?>
<conductor_redundancy_group>
  <vip>10.4.200.200</vip>
  <vrid>2</vrid>
  <name>Conductor</name>
</conductor_redundancy_group>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<conductor_redundancy_groups href="/conductor_redundancy_groups" product="AWS Elemental
Conductor Live" version="3.3.nnnnn">
  <conductor_redundancy_group>
```

```
<id type="integer">1</id>
<vip>10.4.200.200</vip>
<vrid type="integer">2</vrid>
<name>Conductor</name>
<product_name>conductor_live</conductor_live>
</conductor_redundancy_group>
</conductor_redundancy_groups>
```

PUT: Modify a Conductor Redundancy Group

Change the name, VIP or VRID of the specified Conductor redundancy group.

HTTP Request and Response

Request URL

```
PUT http://<Conductor IP address>/conductor_redundancy_groups/<ID of Conductor
redundancy group>
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called "Header Content for User Authentication"](#).

Request Body

The body contains only the elements to change: name, VIP or VRID. For the format and other elements, see [the section called "POST: Create a Redundancy Group"](#).

Example

This request changes the name of the Conductor redundancy group with the ID 1. It changes the name to RedundancyB.

```
PUT http://198.51.100.0/conductor_redundancy_groups/1
-----
```

```
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<conductor_redundancy_group>
  <name>RedundancyB</name>
</conductor_redundancy_group>
```

GET: Get the Attributes of the Conductor Redundancy Group

Get a list of all Conductor redundancy groups, including the attributes of each group. There is only ever one Conductor redundancy group, so the list contains only one item. However, you cannot assume its ID is always going to be 1: if you create a Conductor redundancy group and then delete it and then create another one, the ID of the new group will be 2.

There is only one Conductor redundancy group, so there is no difference between GET and GET List; only GET is described in this guide.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/conductor_redundancy_groups
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response is XML content consisting of one conductor_redundancy_groups element with the following.

- An HREF attribute that specifies the product and version installed on the Conductor Live node.
- Zero or one conductor_redundancy_group elements. Each element contains several elements.

Element	Value	Description
id	Integer	The ID for this Conductor redundancy group.
vip	String	A valid IPv4 address. This address is the “cluster ID” for the two Conductor nodes.
vrid	Integer	The Virtual Router ID, a number from 1-254.
enabled	Boolean	True if Conductor redundancy is enabled on the cluster. False if it is not enabled. Read only.
name	String	The name of the group.
product_name	String	This setting is always “conductor_live.” Read -only.

Example

The response to this request shows one Conductor redundancy group with the ID 1.

```
GET http://198.51.100.0/conductor_redundancy_groups
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
-----
<?xml version="1.0" encoding="UTF-8"?>
<conductor_redundancy_groups href="/conductor_redundancy_groups" product="AWS Elemental
Conductor Live" version="3.3.43776">
  <conductor_redundancy_group>
    <id type="integer">2</id>
    <vip>10.10.10.111</vip>
```

```
<vrid type="integer">13</vrid>
<enabled type="boolean">>false</enabled>
<name>ConductorRedundancy</name>
<product_name>conductor_live</product_name>
</conductor_redundancy_group>
</conductor_redundancy_groups>
```

DELETE: Delete a Conductor Redundancy Group

Delete the Conductor redundancy group that has the specified ID. The group must first be [disabled](#).

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/conductor_redundancy_groups/<ID of Conductor
redundancy group>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Example

This request deletes the Conductor redundancy group with the ID 1.

```
DELETE http://198.51.100.0/conductor_redundancy_groups/1
```

POST Enable: Enable Conductor Redundancy Group

Enable Conductor Redundancy mode (high availability) on the two Conductor nodes in the Conductor redundancy group in order to enable the failover mode for the Conductor nodes.

If you are using a VM, take a snapshot before enabling high availability. See the VMware VSphere help text for more information.

HTTP Request and Response

Request URL

```
POST http://<IP of primary Conductor node>/conductor_redundancy_groups/<ID of Conductor redundancy group>/enable
```

where:

- <ID of primary Conductor node> is the ID of the node that you want to have the role of primary once redundancy is enabled. In other words, perform this command from the node that you want to become the primary Conductor.

The process of enabling Conductor redundancy takes more than a few minutes, so you should only enable when you are absolutely sure you have performed all the necessary cluster configuration. While the mode is being enabled, the Conductor nodes cannot take REST API requests.

Once Conductor redundancy is enabled, you must send API requests to the virtual IP address (VIP) you specified in the [POST Conductor Redundancy Group](#). The virtual IP redirects the request to the Conductor node that is currently primary.

So the process is:

1. To enable Conductor redundancy, send the POST Enable request to the Conductor node that is currently primary.
2. Once Conductor redundancy is successfully enabled, send all future API commands to the virtual IP address.

Warning

When Conductor redundancy is enabled, do not send requests to the individual primary Conductor node, even if you know which node is currently the primary.

Call Header

- Accept: Set to `application/xml`
- Content-Type: Set to `application/xml`

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response is returned immediately (the system does not wait for Conductor redundancy to finish). The response is identical to the [response for GET List of Redundancy Groups](#).

Example

This example enables Conductor redundancy on the Conductor redundancy group (which has ID 1) by submitting a request from the node 198.51.100.0, which is currently the primary node.

```
POST http://198.51.100.0/conductor_redundancy_groups/1/enable
```

DELETE Disable: Disable Conductor Redundancy Group

Disable Conductor Redundancy mode (high availability) on the two Conductor nodes in the Conductor redundancy group.

If you are using a VM, take a snapshot before disabling high availability. See the VMware VSphere help text for more information.

HTTP Request and Response

Request URL

```
DELETE http://<VIP>/conductor_redundancy_groups/<ID of Conductor redundancy group>/  
disable
```

The process of disabling Conductor redundancy takes more than a few minutes, so you should not disable it unless you have a good reason:

- You must disable Conductor redundancy to change any of the network settings.
- You must disable Conductor redundancy in order to upgrade the Conductor Live software.

Note

You do not need to disable Conductor redundancy in order to add or remove worker nodes from the cluster!

While the mode is being disabled, the Conductor nodes cannot take REST API requests.

Once Conductor redundancy is disabled, you must send API requests to the IP address of the Conductor node that is currently primary.

So the process is:

1. To disable Conductor redundancy, send the POST Enable request to the virtual IP address.
2. Once Conductor redundancy is successfully enabled, send all future API commands to the Conductor node that is currently primary.

Warning

When Conductor redundancy is disabled, do not send requests to the VIP.

Call Header

- Accept: Set to `application/xml`

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response is identical to the [response for GET List of Redundancy Groups](#).

Example

This example disables Conductor redundancy on the Conductor redundancy group (which has ID 1) by submitting a request from the VIP:

```
DELETE http://10.4.200.200/conductor_redundancy_groups/1/disable
```

Setting Up Members of a Conductor Redundancy Group

Topics

- [POST: Add a Node to a Conductor Redundancy Groups](#)
- [PUT: Modify a Member of a Conductor Redundancy Group](#)
- [GET List: Get a List of Conductor Redundancy Group Members](#)
- [GET: Get the Attributes of a Conductor Redundancy Group Member](#)
- [DELETE: Remove a Node from the Conductor Redundancy Group](#)

POST: Add a Node to a Conductor Redundancy Groups

Add a Conductor node to the specified Conductor redundancy group. When setting up a Conductor redundancy group, you must add exactly two nodes.

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/conductor_redundancy_groups/<ID of Conductor
redundancy group>/members
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The response contains XML content consisting of the following.

- One `conductor_redundancy_groups` element that contains:
 - One or two `conductor_redundancy_group_member` elements with the following elements.

Element	Value	Description
node_id		ID of the Conductor node

Response

The response repeats back the data that you posted with the addition of the following.

- id: The newly assigned ID for the member.

Example

Request

Add the node with the ID 2 to the Conductor redundancy group. In this example, the Conductor redundancy group has the ID 1.

```
POST http://198.51.100.0/conductor_redundancy_groups/1/members
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<conductor_redundancy_group_member>
<node_id>2</node_id>
</conductor_redundancy_group_member>
```

Response

The response shows the “Conductor redundancy group member” data and the standard node data that was applied to this node. Note that the node has the member ID of 3 (its ID in the Conductor redundancy group) but has the node ID of 2 (its ID in the cluster). For details, see [the section called “GET List: Get a List of Redundancy Group Members”](#).

```
<?xml version="1.0" encoding="UTF-8"?>
<conductor_redundancy_group_member href="/conductor_redundancy_groups/3/members"
product="AWS Elemental Conductor Live" version="3.3.nnnnn">
  <id>3</id>
  <node>
    <id>2</id>
```

```
<hostname>elae-12345678</hostname>
<ip_addr>10.4.136.92</ip_addr>
<status>online</status>
<product_name>Live</product_name>
<version>2.7.0.67890</version>
<channels>0</channels>
</node>
</conductor_redundancy_group_member>
```

PUT: Modify a Member of a Conductor Redundancy Group

There is no PUT for members of a Conductor redundancy group. You cannot change any of its attributes.

GET List: Get a List of Conductor Redundancy Group Members

Get the list of the nodes that are members of the specified Conductor redundancy group.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/conductor_redundancy_groups/<ID of Conductor
redundancy group>/members
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response is XML content consisting of one `conductor_redundancy_group_members` element with the following.

- An HREF attribute that specifies the product and version installed on the Conductor node.
- Zero, one or two `conductor_redundancy_group_member` elements, one for each Conductor node found. Each element contains several elements.

Element	Value	Description
id	Integer	The member ID for this node, an ID for this node that is unique within the Conductor redundancy group.
rank	Integer	Each node in the Conductor redundancy group is assigned a random number. This number is used to display nodes on the Conductor web interface.
node	Object	The node information for this member of the Conductor redundancy group; see the section called “POST: Add a Node to the Cluster” .

Example

The response to this request shows that the Conductor redundancy group with the ID 1 has two members – one node that is the backup and one that is active.

```
GET http://198.51.100.0/conductor_redundancy_groups/1/members
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
-----
<?xml version="1.0" encoding="UTF-8"?>
<conductor_redundancy_group_members href="/conductor_redundancy_groups/1/members"
product="AWS Elemental Conductor Live" version="3.3.nnnnn">
  <conductor_redundancy_group_member>
    <id>2</id>
    <rank>153</rank>
    <node>
      <id>5</id>
      <hostname>elae-12345678</hostname>
      <ip_addr>10.4.136.90</ip_addr>
```

```
<status>online</status>
<product_name>AWS Elemental Conductor Live</product_name>
<version>3.3.nnnnn</version>
<channels>0</channels>
<inflight_channels>0</inflight_channels>
<mptses>0</mptses>
<active_alerts>0</active_alerts>
<recent_error_messages>0</recent_error_messages>
<redundancy_group nil='true'>
  <authentication>
    <user_id>1</user_id>
  </authentication>
</node>
</conductor_redundancy_group_member>
<conductor_redundancy_group_member>
  <id>8</id>
  <node>
    <id>2</id>
    <hostname>elae_33445566</hostname>
    <ip_addr>10.4.136.91</ip_addr>
    <status>online</status>
    <product_name>Live</product_name>
    <version>2.7.0.123456</version>
    .
    .
    .
  </node>
</conductor_redundancy_group_member>
</conductor_redundancy_group_members>
```

GET: Get the Attributes of a Conductor Redundancy Group Member

Get the attributes of the specified member in the specified Conductor redundancy group.

HTTP Request and Response

Request URL

```
GET http://<Conductor IP address>/conductor_redundancy_groups/<ID of Conductor
redundancy group>/members/<ID of member node>
```

Call Header

- Accept: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Response

The response is XML content consisting of one `conductor_redundancy_group` element with the same elements as the response for GET Member List above.

Example

This request gets the attributes for the Conductor redundancy group that has the ID 1.

```
GET http://198.51.100.0/conductor_redundancy_groups/1/members/1
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
-----
<?xml version="1.0" encoding="UTF-8"?>
<conductor_redundancy_group_member href="/conductor_redundancy_groups/1/members"
  product="AWS Elemental Conductor Live" version="3.3.nnnnn">
  <id>1</id>
  <rank>13</rank>
  <node>
    .
    .
    .
  </node>
</conductor_redundancy_group_member>
```

DELETE: Remove a Node from the Conductor Redundancy Group

Remove the Conductor node with the specified member ID from the Conductor redundancy group. Conductor redundancy must be disabled; if it is currently enabled, disable it as described in [the section called “DELETE Disable: Disable Conductor Redundancy Group”](#).

```
DELETE http://<Conductor IP address>/conductor_redundancy_groups/<ID of Conductor
  redundancy group>/members/<ID of member node>
```

Backing Up the Conductor Database

Topics

- [PUT: Modify Database Backup Settings](#)
- [POST: Backup Database Now](#)

PUT: Modify Database Backup Settings

You can view and modify the current backup settings for the conductor database on the web UI at Settings>Backups. The following command allows you to modify these settings via REST.

HTTP Request and Response

Request URL

```
PUT http://<Conductor IP address>/cluster/backups
```

Call Header

- Accept: Set to application/xml
- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

The response contains XML content consisting of one backup element with the following elements.

Element	Value	Description
database_backup_dir	string	The directory where backup files are stored. Default: /home/elemental/database_backups

Element	Value	Description
database_backup_count	integer	The number of backups stored in the system. Default: 5
database_backup_minutes	integer	The time , in minutes, between automatic backups. Default: 1440

Example

This request changes the backup directory to /home/elemental/database_backups, sets the system to retain the 6 most recent backups and sets the time between backups to three hours.

```
PUT http://198.51.100.0/cluster/backups
-----
Content-type:application/vnd.elemental+xml;version=3.3.0
Accept:application/xml
-----
<?xml version="1.0" encoding="UTF-8"?>
<backup>
  <database_backup_dir>/home/elemental/database_backups2</database_backup_dir>
  <database_backup_count>6</database_backup_ count>
  <database_backup_minutes>180</database_backup_minutes>
</backup>
```

POST: Backup Database Now

To initiate an immediate database backup, send the following command.

HTTP Request and Response

Request URL

```
POST http://<Conductor IP address>/cluster/backups
```

Call Header

- Accept: Set to application/xml

- Content-Type: Set to application/xml

If you are implementing user authentication, you must also include three authorization headers; see [the section called “Header Content for User Authentication”](#).

Request Body

There is no body in the backups command.

Document History for Conductor Live API Guide

The following table describes the release history of this guide.

- **API version:** 3.25
- **Release notes:** [current Release Notes](#)

The following table describes the documentation for this release of Conductor Live. For notification about updates to this documentation, you can subscribe to an RSS feed.