API Reference

AWS Elemental Delta

Version 2.3

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Elemental Delta: API Reference

Copyright © 2024 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

xi

API Changes since AWS Elemental Delta 2.2 ... 1

Input Filter API Changes ... 1

HTTP PUT, MSS, and WebDAV Input Changes ... 1

VOD Catalog Input Filter Changes ... 2

Output Filter API Changes ... 2

DASH Package Output Changes ... 2

DRM Output Filter Changes ... 3

VOD Catalog Contents API Changes ... 5

Configuration and Cluster Settings API Changes ... 6

AWS Credentials API Changes ... 6

Mount Points API Changes ... 6

Nodes API Changes ... 6

Working with the API ... 7

The API Protocol ... 7

Concepts and Terminology ... 7

HTTP Requests ... 8

Encoding HTTP Requests ... 10

Case-Sensitivity of Delta Attributes ... 10

Boolean Values for Delta Attributes ... 10

Location Type Elements ... 11

Server Type Elements ... 11

Null Values for Delta Attributes ... 11

Using the API with User Authentication Enabled ... 12

Clean GET Requests ... 13

Slim GET Requests ... 16

HTTP Responses ... 17

IDs of Entities ... 18

Commands and Filters ... 20

List of Commands ... 20

Input Filters ... 20

Remote Input Contents ... 21

Input Users ... 21

Contents ... 22

Output Filters in a Content ... 23
Tracks ... 23
User Agent Presets ... 24
Output Templates ... 24
VOD Catalog Output Templates ... 25
Alerts and Messages ... 25
Settings ... 26
Cluster Configuration ... 26
VIPs ... 26
List of Output Filters ... 27
Ingesting Content ... 31
Working with Input Filters ... 31
PUT Input Filter ... 31
GET Input Filters List ... 32
GET Input Filter ... 34
DELETE Input Filter ... 35
Working with Input Users ... 36
POST Input User ... 36
GET Input Users List ... 37
GET Input User ... 38
DELETE Input User ... 39
Working with a Live or VOD Contents Entity ... 39
Content Entities and Output Filters ... 40
Content Aliases ... 41
POST Remote Input Content ... 43
PUT Remote Input Content ... 45
GET Remote Content ... 47
GET Remote Content for an Input Filter ... 47
GET Contents List ... 51
GET a Content and Filters ... 59
GET a Content and Tracks ... 61
PUT Content ... 62
DELETE a Content Entity ... 64
Working with a VOD Catalog Contents Entity ... 65
POST VOD Catalog - Initiate Ingest ... 67
POST VOD Catalog Content ... 68

DELETE a Content Metadata ... 70
PUT VOD Catalog Contents ... 71
GET VOD Catalog Contents List ... 74
GET a VOD Catalog Content ... 77
DELETE a VOD Catalog Content Entity ... 77
Creating Input Filters ... 79
POST: Create an HTTP PUT Input Filter ... 79
The `input_filter` Element ... 80
The HTTP PUT `filter_settings` Element ... 82
POST: Create an MSS Input Filter ... 92
The `input_filter` Element ... 94
The MSS `filter_settings` Element ... 95
POST: Create a Remote Input Filter ... 104
The `input_filter` Element ... 106
The Remote Input `filter_settings` Element ... 107
POST: Create an RTMP Input Filter ... 113
The `input_filter` Element ... 114
The RTMP `filter_settings` Element ... 116
POST: Create a UDP Input Filter ... 126
The `input_filter` Element ... 128
The UDP `filter_settings` Element ... 130
POST: Create a Watch Folder Input Filter ... 141
The `input_filter` Element ... 142
The Watch Folder `filter_settings` Element ... 143
POST: Create a WebDAV Input Filter ... 145
The `input_filter` Element ... 146
The WebDAV Folder `filter_settings` Element ... 148
POST: Create a VOD Catalog Input Filter ... 158
The `input_filter` Element ... 160
The VOD Catalog Folder `filter_settings` Element ... 161
Outputting Content ... 163
Creating Output Filters ... 163
Methods for Creating Output Filters ... 163
Modifying Output Filters ... 165
Modifying an Element ... 166
Adding an Element ... 167

Removing an Element ... 168
Modify Tags in an Output Filter ... 169
Modify Tags in a `filter_settings` ... 170
Add a New stream_set ... 172
Add a Track to a `stream_set` ... 175
Modify Tags in a Track ... 177
Modify Tags in a `keyprovider_settings` ... 180
Add a `nonce_key_pairs` ... 181
Add a `nonce_key_pair` ... 183
Viewing Filters List: GET Output Filters List ... 184
Viewing a Live or VOD Filter: GET Output Filter ... 186
HTTP Request and Response ... 186
Viewing VOD Catalog Endpoints: GET Endpoints ... 187
Deleting an Output Filter: DELETE ... 188
Creating Package Output Filters ... 190
PUT: Create a CMAF Package Output Filter ... 190
The `filter` Element ... 195
The `filter_settings` Element ... 199
The `stream_set` Element ... 208
PUT: Create a DASH-ISO Package Output Filter ... 213
The `filter`Element ... 217
The `filter_settings` Element ... 220
The `stream_set` Element ... 222
PUT: Create an HDS Package Output Filter ... 224
The `filter` Element ... 228
The `filter_settings` Element ... 232
The `stream_set` Element ... 237
PUT: Create an HLS Package Output Filter ... 239
The `filter` Element ... 246
The `filter_settings` Element ... 250
The `stream_set` Element ... 259
PUT: Create an MP4 Package Output Filter ... 264
The `filter` Element ... 268
The `filter_settings` Element ... 272
The `stream_set` Element ... 273
PUT: Create an MPEG-TS Package Output Filter ... 275

The `filter` Element ... 279
The `filter_settings`Element ... 283
The `stream_set` Element ... 285
PUT: Create an MSS Package Output Filter ... 287
The `filter` Element ... 291
The `filter_settings` Element ... 294
The `stream_set` Element ... 295
PUT: Create a Passthrough Package Output Filter ... 298
The `filter` Element ... 300
Working with Tracks: GET Tracks ... 303
The `track` Element for Audio ... 304
The `track` Element for Subtitles ... 305
The `track` Element for Video ... 306
Creating Access Restriction Output Filters ... 308
PUT: Create a DRM Output Filter ... 308
Common Encryption (CENC) Output Filter ... 309
Flash Access Output Filter ... 322
HLS Encryption Output Filter ... 332
PlayReady Output Filter ... 361
PUT: Create an Authentication Output Filter ... 375
Akamai G2o Output Filter ... 376
Cisco URL Signing Output Filter ... 385
PUT: Create a Whitelist or Blacklist Output Filter ... 393
The IP Whitelist and IB Blacklist `filter` Element ... 395
The IP Whitelist and IP Blacklist `filter_settings` Element ... 398
PUT: Create a User Agent Output Filter ... 399
The User Agent `filter` Element ... 402
The User Agent `filter_settings` Element ... 405
Working with User Agent Presets ... 406
POST User Agent Preset ... 407
GET User Agent Presets List ... 408
DELETE User Agent Preset ... 410
Creating Processing Output Filters ... 411
PUT: Create an Ad Removal Output Filter ... 411
of The Ad Removal `filter` Element ... 412
The Ad Removal `filter_settings` Element ... 416

PUT: Create an Ad Replace Output Filter ... 416
The Ad Replace `filter`Element ... 418
The Ad Replace `filter_settings` Element ... 421
PUT: Create a Bitrate Selector Output Filter ... 424
The Bitrate Selector `filter` Element ... 425
The Bitrate Selector `filter_settings` Element ... 429
PUT: Create a Blackout Output Filter ... 430
The Blackout `filter` Element ... 431
The Blackout `filter_settings` Element ... 435
PUT: Create a Cache Settings Output Filter ... 436
The Cache Settings `filter` Element ... 437
The Cache Settings `filter_settings` Element ... 441
PUT: Create a File Copy Output Filter ... 442
The File Copy `filter` Element ... 443
The File Copy `filter_settings` Element ... 447
PUT: Create a Live to VOD Output Filter ... 448
The Live to VOD `filter` Element ... 451
The Live to VOD `filter_settings`Element ... 454
PUT: Create a Live to VOD Catalog Output Filter ... 460
The Live to VOD Catalog `filter` Element ... 462
The Live to VOD Catalog `filter_settings` Element ... 465
PUT: Create a Preroll Ad Insertion or Postroll Ad Insertion Output Filter ... 469
The Preroll Ad Insertion and Postroll Ad Insertion `filter` Element ... 470
The Preroll Ad Insertion and Postroll Ad Insertion `filter_settings` Element ... 474
PUT: Create a Time Delay Output Filter ... 477
The Time Delay `filter` Element ... 478
The Time Delay `filter_settings` Element ... 482
PUT: Create a VOD Clip Output Filter ... 482
The VOD Clip `filter` Element ... 483
The VOD Clip `filter_settings` Element ... 487
Working with Filter Sets and Add Filters ... 492
Filter Sets ... 492
Example 1 ... 496
Example 2 ... 497
Example 3 ... 499
Example 4 ... 500

PUT Add_Filters 501
Working with Output Templates 503
POST Output Template 503
The filter Element 504
The filter_settings Element 505
Stream Sets and Track Selectors 505
PUT Output Template 512
GET Output Templates List 512
GET an Output Template 513
DELETE an Output Template 513
Monitoring Delta 514
GET Alerts 514
GET Messages 516
Healthz: Get Node Health 518
Configuring Delta 519
Querying Settings 519
GET Settings 519
GET General Settings 522
GET SNMP Configuration 523
Managing Amazon S3 Credentials 524
POST Amazon S3 Credentials 524
PUT Amazon S3 Credentials 525
GET Amazon S3 Credentials 526
DELETE Amazon S3 Credentials 527
Configuring Mount Points 527
POST Mount Point 527
GET Mount Point List 530
DELETE Mount Point 531
Configuring an AWS Elemental Delta Cluster 531
GET: Query the Cluster Configuration 532
GET: Query Pave Progress 536
PUT: Modify the Cluster 536
POST: Create a Virtual IP Address (VIP) 537
PUT: Modify a VIP 539
GET VIPs List 540
GET a VIP 541

DELETE a VIP 542
Working with Authorized Users 542
POST: Add an Authorized User 542
GET Users 544
DELETE a User 546
XML Structure 547
Package Output Filter 547
DRM Output Filter 548
Authentication Output Filters 549
Processing Output Filters 550
Document History 551

This is version 2.3 of the AWS Elemental Delta documentation. This is the latest version. For prior versions, see the *Previous Versions* section of AWS Elemental Delta Documentation.

For Upgrading Users: API Changes since AWS Elemental Delta 2.2

The Delta 2.3 release introduced API changes to support new features and generally improve the API.

Commands using older Delta API XML versions might not be compatible with these changes. To ensure commands are processed if you're using an older API version, use versioned endpoints as described in API Version Prefix.

The following sections describe the changes.

Topics

- Input Filter API Changes
- Output Filter API Changes
- VOD Catalog Contents API Changes
- Configuration and Cluster Settings API Changes

Input Filter API Changes

The following sections describe the API changes to input filters.

Topics

- HTTP PUT, MSS, and WebDAV Input Changes
- VOD Catalog Input Filter Changes

HTTP PUT, MSS, and WebDAV Input Changes

To allow for content encryption at rest, HTTP PUT, MSS, and WebDAV input filters now include an `encrypt_storage` Boolean tag in the `filter_settings` object. This tag is optional. The default if not specified is `false`.

```
<input_filter>
  <filter_type>http_put_input</filter_type>
```

```
  <filter_settings>
    <content_window_type>keep_seconds</content_window_type>
    <seconds_to_keep>14400</seconds_to_keep>
    <vod_content>false</vod_content>
    <automatically_promote_after_ingest>false</automatically_promote_after_ingest>
    <promotion_delay_seconds>1800</promotion_delay_seconds>
    <vod_output_template_alias>vcat</vod_output_template_alias>
    <time_source>ingest_time</time_source>
    <encrypt_storage>true</encrypt_storage>
  </filter_settings>
</input_filter>
```

VOD Catalog Input Filter Changes

To allow for content encryption at rest, VOD Catalog input filters now include an `encrypt_path` string tag in the `filter_settings` object. This tag is optional.

```
<input_filter>
  <filter_type>vod_catalog_input</filter_type>
  <filter_settings>
    <location>/data/server/cores/</location>
    <vod_output_template_alias>vcat</vod_output_template_alias>
    <vod_url_prefix>sports</vod_url_prefix>
    <take_ownership_of_content>false</take_ownership_of_content>
    <encrypt_path></encrypt_path>
  </filter_settings>
</input_filter>
```

Output Filter API Changes

The following sections describe the API changes to output filters.

Topics

- DASH Package Output Changes
- DRM Output Filter Changes

DASH Package Output Changes

The following changes were made to tags in the DASH-ISO output filter `filter_settings` object:

- To support multi-period DASH manifests, the `ad_avail` tag has been replaced with a `multiperiod` Boolean tag. The default if not specified is `false` (the entire manifest is included in a single period).
- The default value for `min_update_period` is now 15 seconds (from 30).

```
<filter>
  <filter_type>dash_iso_package</filter_type>
  <endpoint>false</endpoint>
  <use_default_stream_sets>true</use_default_stream_sets>
  <filter_settings>
    <fragment_duration>2</fragment_duration>
    <index_duration>60</index_duration>
    <multiperiod>false</multiperiod>
    <min_update_period>15</min_update_period>
    <min_buffer_time>30</min_buffer_time>
    <suggested_presentation_delay>25</suggested_presentation_delay>
  </filter_settings>
  <stream_sets/>
</filter>
```

DRM Output Filter Changes

The following changes were made to tags in DRM output filters:

- In Common Encryption, HLS Encryption, and PlayReady DRM output filters, `speke` is now available as a `keyprovider_type`. For more information, see the Secure Packager and Encoder Key Exchange Documentation . For required tags when you use SPEKE as a key provider, see PUT: Create a DRM Output Filter.

```
<filter>
  <filter_type>hls_encryption</filter_type>
  <filter_settings>
    <encryption_type>AES-128</encryption_type>
    <key_rotation_count>3</key_rotation_count>
    <iv_follows_segment_number>true</iv_follows_segment_number>
    <key_id></key_id>
    <keyprovider_type>speke</keyprovider_type>
    <repeat_ext_x_key>false</repeat_ext_x_key>
    <keyprovider_settings>
      <system_ids>
```

```
        <system_id>00000000-1234-5678-1111-000000000000</system_id>
      </system_ids>
      <content_id>cbdhsah</content_id>
      <speke_server>
        <uri>https://speke.uri</uri>
        <username>user</username>
        <password>password</password>
      </speke_server>
    </keyprovider_settings>
  </filter_settings>
  <stream_sets/>
</filter>
```

- In the HLS Encryption DRM output filter, you use **`Irdeto`** as the key provider, `ca_key_server` is now available as a `key_source`. This option indicates that you're using the Irdeto CA protection type.

```
<filter href="/contents/1/filters/6" product="Delta" version="2.3.0.201116">
  <filter_type>hls_encryption</filter_type>
  <filter_settings>
    <encryption_type>AES-128</encryption_type>
    <key_rotation_count>3</key_rotation_count>
    <iv_follows_segment_number>true</iv_follows_segment_number>
    <keyprovider_type>irdeto</keyprovider_type>
    <repeat_ext_x_key>false</repeat_ext_x_key>
    <keyprovider_settings>
      <content_id>cbdhsah</content_id>
      <account_id>0123456789</account_id>
      <use_https>false</use_https>
      <content_key>generate_new_key</content_key>
      <key_source>ca_key_server</key_source>
      <request_cenc_key>false</request_cenc_key>
      <request_playready_key>false</request_playready_key>
      <kms_username>kmsusername</kms_username>
      <kms_password>kmspword1</kms_password>
      <service_url>
        <uri>http://service.url</uri>
        <username>uname</username>
        <password>pword</password>
      </service_url>
    </keyprovider_settings>
  </filter_settings>
  <stream_sets/>
```

```
</filter>
```

VOD Catalog Contents API Changes

When you have many VOD catalog contents and send a GET request, append `?slim=true` to the request to retrieve results more quickly. This way, the response includes information about the content itself and not the filters applied to it.

```
GET http://<Delta IP address>:8080/vod_catalog_contents?slim=true
-------------------------------------------------------------------------------
<vod_catalog_contents>
  <vod_catalog_content>
    <name>HLScontent</name>
    <path>default_HLS/vod_106</path>
    <content_alias nil="true"/>
    <resource_id></resource_id>
    <resource_id_alt></resource_id_alt>
    <delta_owned>true</delta_owned>
    <vod_output_template_alias>marisol</vod_output_template_alias>
    <encrypted>false</encrypted>
    <created_at>2018-10-03T10:47:18-07:00</created_at>
    <updated_at>2018-10-03T10:47:18-07:00</updated_at>
  </vod_catalog_content>
  <vod_catalog_content>
    <name>UDPcontent</name>
    <path>UDP5/vod_112</path>
    <content_alias nil="true"/>
    <resource_id></resource_id>
    <resource_id_alt></resource_id_alt>
    <delta_owned>true</delta_owned>
    <vod_output_template_alias>marisol</vod_output_template_alias>
    <encrypted>false</encrypted>
    <created_at>2018-10-03T10:47:44-07:00</created_at>
    <updated_at>2018-10-03T10:47:44-07:00</updated_at>
  </vod_catalog_content>
</vod_catalog_contents>
```

Configuration and Cluster Settings API Changes

The following sections describe the API changes to general configuration settings and cluster information in AWS Elemental Delta.

Topics

- AWS Credentials API Changes
- Mount Points API Changes
- Nodes API Changes

AWS Credentials API Changes

To improve security, the response to a `GET ..aws_credentials` no longer includes the `secret_key`.

Mount Points API Changes

To improve security, the response to a `GET ..mount_pointss` no longer includes the `password`.

Nodes API Changes

When AWS Elemental Delta performs a failover, you can now see the pave progress of the new leader node. The following request returns a hash that includes the status of the pave.

```
GET http://<Delta IP address>:8080/nodes/<node_ID>.json
```

Basics of the AWS Elemental Delta REST API

This guide describes the actions that you can perform in the Delta REST API.

The following sections provide basic getting started information about the API.

Topics

- The API Protocol
- Concepts and Terminology
- HTTP Requests
- Using the API with User Authentication Enabled
- Clean GET Requests
- Slim GET Requests
- HTTP Responses
- IDs of Entities

The API Protocol

You can access the AWS Elemental Delta API using HTTP. The API follows the Representational State Transfer (REST) architectural framework. In accordance with REST guidelines, the API exposes four types of operations based on which HTTP method you use to make a request:

- POST
- GET
- PUT
- DELETE

Concepts and Terminology

This section defines some common concepts and terminology used in this guide.

Entity

An *entity* is an object that you can use the REST API to take action upon. These are the entities in the Delta API:

- `input_filters`
- `contents`
- `remote_input_contents`
- `output filters`
- `output_templates`
- `input_users`
- `settings`
- `alerts`
- `messages`
- `clusters`

Attribute

An *attribute* is a setting of an entity. For example, the name of an input filter is an attribute.

In the Delta API, these attributes are passed in the XML body of an HTTP request or response.

Tag

A *tag* is the XML representation of an attribute. For example, the name attribute of an input filter is represented by the value of the `name` tag.

Element

An *element* is a grouping related tags. For example, `filter_settings` on an output filter is an element. All of the tags within this element relate to settings specific to this filter. You can't directly perform an action on an element.

HTTP Requests

HTTP requests consist of an HTTP URL, a header, and a body.

HTTP URL Content

The content of the HTTP URL consists of the selected operation, the URL of the Delta node, and specific parameters. For example:

```
PUT http://10.24.34.2:8080/input_filter/InputFilterID
```

API Version Prefix for Endpoints in REST Requests

For best results, whether you submit REST requests manually or from an automation system, use an API version prefix for all endpoints. This prefix allows you to specify to Delta which API version to use to interpret your data.

> **Warning**
>
> Delta always forms responses according to the currently installed software and API version, regardless of the version prefix used in the request. This is why, if you want to use another API version, you must specify the prefix for it instead of the one for the currently installed version.

In the following example, the request is sent to the `/contents` endpoint where the server interprets the data as compatible with Delta API version 2.0.1.

```
POST http://Delta IP address:8080/api/v2.0.1/contents
```

Although it is not recommended, you can omit the version prefix from the request. In this case, the server interprets the data as compatible with the currently installed version. If you use tags from a different version, you must also use the appropriate version prefix or the system will reject your request.

Header Content – Standard Elements

The header content must always contain two standard elements:

- Content-Type: set to `application/xml`
- Accept: set to `application/xml` (for PUT and POST requests only.)

Header Content Requirements When Configuring for User Authentication

If you configure your Delta deployment for user authentication (meaning that users must log into Delta), then the header must also include:

- `X-Auth-User` header.
- `X-Auth-Expires` header (optional).
- `X-Auth-Key` header includes the API key unique to each user.

Body Content Requirements for HTTP Requests

The body of the HTTP requests consists of XML content. The body is not needed for GET or DELETE requests. You must include a body as described here:

- Most POST requests, such as when you create an input filter.
- All PUT requests, such as when you make a change to an existing input filter.

See the topic for the action that you are performing to determine what tags to include in the XML body.

Topics

- Encoding HTTP Requests
- Case-Sensitivity of Delta Attributes
- Boolean Values for Delta Attributes
- Location Type Elements
- Server Type Elements
- Null Values for Delta Attributes

Encoding HTTP Requests

HTTP requests must be encoded in the following ways:

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

Case-Sensitivity of Delta Attributes

The names and the values of all Delta attributes are case-sensitive.

Boolean Values for Delta Attributes

Boolean values in Delta attributes must be entered as **`true`** or **`false`**. Delta does not accept **`0`** or **`1`** for boolean values.

Location Type Elements

A *location* type element holds tags for a URI or path, username, and password. Use this type when you might or might not need user credentials to access the resource.

A location type element always requires the URI. The username and password for the URI may be required or optional.

Example

```
<incoming>
   <username>special_user</username>
   <password>secret</password>
   <uri>10.24.34.2</uri>
  </incoming>
```

Server Type Elements

A *server* type element holds tags for the URI or path of a server and the username and password for that server. Use this type when your access to the resource requires user credentials and other data.

All of the tags in the following example are required for a server type element.

Example

```
<server>
   <api_key>special_key</api_key>
   <api_secret>secret</api_secret>
   <uri>10.24.34.2</uri>
  </server>
```

Null Values for Delta Attributes

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

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

Using the API with User Authentication Enabled

When user authentication is enabled, users have to pass valid credentials to access the Delta nodes. HTTP requests must also include the headers described in the following table.

To enable authentication, confer with the person who performed the initial cluster configuration or see the AWS Elemental Delta version 2.3 Configuration Guide

If you are using API with user authentication enabled, then the header of each request must include the following headers.

Header	Description
`X-Auth-User`	The username of the user using the API. **Note** The user's password is not included in the header.
`X-Auth-Expires`	The date and time when 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 synch, you may want to make adjustments to accommoda te the difference.
`X-Auth-Key`	An MD5 hash of the API key for the API user. An administrator generates this key as follows: 1. Log on via the web interface and go to Settings > User Profile. 2. Click the Reset API Key (key icon) for the applicable users.

Header	Description
	3. Provide the individual user with a key, for example, via email. For information on hashing the API key, see below.

Hashing the API Key

To construct the `X-Auth-Key` header, hash the API key as follows:

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

where:

- The + operator indicates string concatenation without any delimiters.
- Each parameter in this expression is entered as a string.
- The `url` parameter is the path portion of the request URL minus query parameters and without an API version prefix. The URL must not have a slash at the end.

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

AuthCurl Scripts

Two authentication helper (AuthCurl) scripts are available to help construct these headers:

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

These scripts are in:

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

Clean GET Requests

The response to a clean GET request includes only the tags that are necessary to construct the body of a POST request. You can send a GET request on an existing entity and include the `clean` parameter in the request. The response:

- Omits the `id` tags and other tags that do not apply to a POST.
- Includes tags that may be used in the POST but are dropped when the entity is stored.

To create a POST request from a clean GET response

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

```
GET http://10.24.34.2:8080/contents/2?clean=true
```

2. As needed, modify the values of the tags as they apply to the entity that you're creating. For example, if you're creating a new Watch Folder input filter, you might change the value of the `incoming` tag to show that the new folder is watching a different folder.
3. Paste the revised response in the body of the POST request for the new entity.

The following table describes what tags are returned in the response to a clean GET request.

Tag	In Standard GET (without `clean`)	In Clean GET (with `clean`)	Comment
`parent_id`	Yes	No	This tag identifies the filter that is this filter's parent. It is not included in the response to a clean GET request because you may be able to use `parent_filter` (below) instead. If an existing output filter is the parent of a filter that you're creating, manually add the `parent_id` tag to its POST request.

Tag	In Standard GET (without `clean`)	In Clean GET (with `clean`)	Comment
`default_endpoint_uri`	Yes	No	This tag contains read-only data that is calculated by the system. It is not a tag in the POST, so it is not included in the clean GET response.
`custom_endpoint_uri`	Yes	No	This tag contains read-only data that is calculated by the system. It is not a tag in the POST, so it is not included in the clean GET response.
`name`	No	Yes	This is a placeholder tag. Use it in a filter set to provide a temporary identifier if the output filter you're creating will be the parent of another filter. For more information about filter sets, see Working with Filter Sets and Add Filters in AWS Elemental Delta.

Tag	In Standard GET (without `clean`)	In Clean GET (with `clean`)	Comment
`parent_filter`	No	Yes	This is a placeholder tag. Use it when you create a filter set (as described in Working with Filter Sets and Add Filters in AWS Elemental Delta) and provide a temporary name to allow one new (not yet created) filter to be associate d with another new (not yet created) filter. This is a placehold er tag. Use it in a filter set to make the new filter a parent of another filter that you're creating. For more information about filter sets, see Working with Filter Sets and Add Filters in AWS Elemental Delta.

Slim GET Requests

A slim GET request limits the amount of information that's returned in the response. When you have many filters on your contents entities, responses to your GET requests can take a considerable

amount of time. Sending slim requests can decrease your wait times. Use slim requests in the following ways:

- Appended to a GET request for a specific live or VOD content ID:
  - The response to the following example includes information about the contents object only. There is no information about the filters.

    ```
    GET http://Delta IP address:8080/contents/<content_ID>?slim=true
    ```

  - The response to the following example includes information about the contents and filter objects only. There is no information about the `filter_settings`, `stream_sets`, or `keyprovider_settings`.

    ```
    GET http://Delta IP address:8080/contents/<content_ID>?slim_filters=true
    ```

- Appended to a GET request for all VOD Catalog contents:
  - The response to the following example includes information about the VOD Catalog contents object only. There is no information about the filters.

    ```
    GET http://Delta IP address:8080/vod_catalog_contents?slim=true
    ```

HTTP Responses

Delta provides an HTTP response to all HTTP PUT, POST, GET, and DELETE requests. Responses consist of a header and a body.

- The header always contains two elements:
  - Content-Type: Set to `application/xml`
  - Accept: For PUT and POST requests only. Set to `application/xml`
- 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 DELETE request: present but empty.
  - Successful GET request: the requested content.

Success Response

If a request is valid, Delta returns the appropriate success response:

- For a POST, PUT or DELETE request: A 200 OK response. The body may contain XML content or may be empty.
- For a GET request: 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 is badly formatted), Delta returns the appropriate HTTP error response, typically an error in the 4xx or 5xx range.
- If the URL of a request is invalid (for example, a tag contains invalid data), then Delta returns a 404 error response.
- If a request is valid but Delta can't fulfill the request for some reason, Delta returns a 422 error response.

IDs of Entities

When you create an entity, Delta automatically assigns it an identifier, which shows as a value of the `id` tag.

These IDs are unique and are typically shown on the Delta web interface under the **ID** column.

You must pass the ID in any POST, GET, and DELETE request. In general, you cannot identify an entity using the name or some other tag. The exception to this is the contents entity, which you can reference by an alias. For more information, see Content Aliases.

Sometimes a set of tags is grouped into an *element*. This element is also assigned a unique ID, but you can't query the element through the API. For example, the `filter_settings` element in a filter entity is assigned an ID, but you can't query that `filter_settings` element directly. You can only query the entities listed in Concepts and Terminology.

Obtaining an ID

The ID is shown in the POST response to a GET List request.

To obtain the ID of an entity

1. Send a GET request to retrieve the list of entities and their IDs for the node.
2. Locate the entity for which you need the ID.

3. Note the value of the `id` tag.

Uniqueness of IDs

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

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

Commands and Filters in AWS Elemental Delta

Topics

- List of Commands
- List of Output Filters

List of Commands

The following sections list the available commands by API resource type.

Topics

- Input Filters
- Remote Input Contents
- Input Users
- Contents
- Output Filters in a Content
- Tracks
- User Agent Presets
- Output Templates
- VOD Catalog Output Templates
- Alerts and Messages
- Settings
- Cluster Configuration
- VIPs

Input Filters

You can perform the following actions on an `input_filter` resource.

Action	Signature	Description
POST	`/input_filters`	Create a new input filter.

Action	Signature	Description
PUT	`/input_filters/<id>`	Modify the specified input filter.
GET	`/input_filters`	Retrieve the list of input filters.
GET	`/input_filters/<id>`	Retrieve the attributes of the specified input filter.
DELETE	`/input_filters/<id>`	Remove the input filter with the specified ID.

Remote Input Contents

You can perform the following actions on a `remote_input_contents` resource.

Action	Signature	Description
POST	`/input_filters/<filter_id>/remote_input_contents`	Create a remote input content managed by the specified input filter.
GET	`/input_filters/<filter_id>/remote_input_contents`	Retrieve the list of remote input content entities managed by the specified input filter.

Input Users

You can perform the following actions on an `input_users` resource.

Action	Signature	Description
POST	`/input_users`	Creates a new input user.

Action	Signature	Description
GET	/input_users	Retrieve the list of input users.
GET	/input_users/<id>	Retrieve the attributes of a specified input user.
GET	/input_users/<id>	Remove an input user with a specified ID.

Contents

You can perform the following actions on contents and vod_catalog_contents resources.

Action	Signature	Description
PUT	/contents/<id>	Modify the specified content entity.
GET	/contents	Retrieve the list of Live and VOD content entities.
GET	/input_users/<id>	Retrieve the attributes of the specified input user.
GET	/contents/<id>	Retrieve the attributes (details, associated output filters, endpoints) of the specified content.
DELETE	/contents/<id>	Remove the content (includin g filters, endpoints, and statistics) for -a specified ID.

Output Filters in a Content

You can perform the following actions on a `filters` resource.

Action	Signature	Description
PUT	`/contents/<id>`	Create a new filter in the specified content. See the next page for a list of output filters.
PUT	`/contents/<id>/add_filters`	Create all or part of a filter tree in the specified content.
PUT	`/contents/<id>/filters/<id>`	Modify the specified filter in the specified content.
GET	`/contents/<id>/filters`	Retrieve the list of filters associated with the specified content.
GET	`/contents/<id>/filters<id>`	Retrieve the attribute of the specified filter that is associated with the specified content.
DELETE	`/contents/<id>/filters/<id>`	Remove the specified filter and all its child filters from the specified content.

Tracks

You can perform the following actions on a `tracks` resource.

Action	Signature	Description
GET	`/contents/<id>/tracks`	Retrieve a list of tracks for the specified content.

User Agent Presets

You can perform the following actions on a `user_agent_presets` resource.

Action	Signature	Description
POST	`/user_agent_presets`	Create a new user agent preset.
GET	`/user_agent_presets`	Retrieve a list of user agent presets.
DELETE	`/user_agent_presets/<id>`	Remove a specified user agent preset.

Output Templates

You can perform the following actions on an `output_templates` resource.

Action	Signature	Description
POST	`/output_templates`	Create a new output template.
GET	`/output_templates`	Retrieve the list of output templates.
GET	`/output_templates/<id>`	Retrieve the specified output template.

Action	Signature	Description
DELETE	`/output_templates/<id>`	Remove the output template with the specified ID.

VOD Catalog Output Templates

You can perform the following actions on a `vod_output_templates` resource.

Action	Signature	Description
POST	`/vod_output_templates`	Create a new VOD Catalog output template.
GET	`/vod_output_templates`	Retrieve the list of VOD Catalog output templates.
GET	`/vod_output_templates/<id>`	Retrieve the specified VOD Catalog output template.
DELETE	`/vod_output_templates/<id>`	Remove the VOD Catalog output template with the specified ID.

Alerts and Messages

You can perform the following actions on an `alerts` or `messages`resource.

Action	Signature	Description
GET	`/alerts`	Retrieve the list of alerts.
GET	`/messages`	Retrieve the list of messages.

Settings

You can perform the following actions on a `settings` resource.

Action	Signature	Description
GET	`/settings`	Retrieve the settings for the Delta cluster.
GET	`/settings/snmp`	Retrieve the SNMP configura tion for the node.

Cluster Configuration

You can perform the following actions on a `clusters` resource.

Action	Signature	Description
GET	`/clusters/1`	Retrieve the configuration attributes of the cluster.
PUT	`/clusters/1`	Modify the configuration of the cluster.

VIPs

You can perform the following actions on a `vips` resource.

Action	Signature	Description
POST	`/clusters/1/vips`	Create a new VIP in the cluster.
PUT	`/clusters/1/vips/<vip_id>`	Modify the attributes of an existing VIP.

Action	Signature	Description
GET	`/clusters/1/vips`	Retrieve the list of VIPs.
GET	`/clusters/1/vips/< vip_id >`	Retrieve the specified VIP.

List of Output Filters

This is a list of output filters available in Delta.

Filter	Filter Type	REST API `filter_type`
Ad Removal	Processing output filter (See PUT: Create an Ad Removal Output Filter)	`ad_removal`
Ad Replace	Processing output filter (See PUT: Create an Ad Replace Output Filter)	`ad_replace`
Akamai G2O Authentication	Authentication output filter (See Akamai G2o Output Filter)	`akamai_g2o_authentication`
Bitrate Selector	Processing output filter (See PUT: Create a Bitrate Selector Output Filter)	`bitrate_selector`
Blackout	Processing output filter (See PUT: Create a Blackout Output Filter)	`blackout`
Cache Settings	Processing output filter	`cache_settings`

Filter	Filter Type	REST API `filter_type`
	(See PUT: Create a Cache Settings Output Filter)	
Cisco URL	Authentication output filter (See Cisco URL Signing Output Filter)	`cisco_url_signing`
CMAF Package	Packaging output filter (See PUT: Create a CMAF Package Output Filter)	`cmaf_package`
Common Encryption	DRM output filter (See Common Encryption (CENC) Output Filter)	`common_encryption`
DASH-ISO Package	Packaging output filter (See PUT: Create a DASH-ISO Package Output Filter)	`dash_iso_package`
File Copy Filter	Processing output filter (See PUT: Create a File Copy Output Filter)	`file_copy`
Flash Access	DRM output filter (See Flash Access Output Filter)	`flash_access`
HDS Package	Packaging output filter (See PUT: Create an HDS Package Output Filter)	`hds_package`

Filter	Filter Type	REST API `filter_type`
HLS Encryption	DRM output filter (See HLS Encryption Output Filter)	`hls_encryption`
HLS Package	Packaging output filter (See PUT: Create an HLS Package Output Filter)	`hls_package`
IP Blacklist	Blacklist output filter (See PUT: Create a Whitelist or Blacklist Output Filter)	`ip_blacklist`
IP Whitelist	Whitelist output filter (See PUT: Create a Whitelist or Blacklist Output Filter)	`ip_whitelist`
Live to VOD Filter	Processing output filter (See PUT: Create a Live to VOD Output Filter)	`live_to_vod`
Live to VOD Catalog Filter	Processing output filter (See PUT: Create a Live to VOD Catalog Output Filter)	`live_to_vod_catalog`
MP4 Package	Packaging output filter (See PUT: Create an MP4 Package Output Filter)	`mp4_package`
MPEG-TS Package	Packaging output filter (See PUT: Create an MPEG-TS Package Output Filter)	`mpeg_ts_package`

Filter	Filter Type	REST API `filter_type`
MSS Package	Packaging output filter (See PUT: Create an MSS Package Output Filter)	`mss_package`
Passthrough	Packaging output filter (See PUT: Create a Passthrough Package Output Filter)	`passthrough`
PlayReady	DRM output filter (See PlayReady Output Filter)	`playready`
Preroll Ad Insertion	Processing output filter (See PUT: Create a Preroll Ad Insertion or Postroll Ad Insertion Output Filter)	`preroll`
Postroll Ad Insertion	Processing output filter (See PUT: Create a Preroll Ad Insertion or Postroll Ad Insertion Output Filter)	`postroll`
Time Delay	Processing output filter (See PUT: Create a Time Delay Output Filter)	`time_delay`
User Agent	User Agent output filter (See PUT: Create a User Agent Output Filter)	`user_agent`
VOD Clip	Processing output filter (See PUT: Create a VOD Clip Output Filter)	`vod_clip`

Ingesting Content

The following sections describe a variety actions that you can take against content that comes in to AWS Elemental Delta.

Topics

- Working with Input Filters
- Working with Input Users
- Working with a Live or VOD Contents Entity
- Working with a VOD Catalog Contents Entity

Working with Input Filters

This section describes the actions that you can take on an input filter that already exists in Delta. For information about creating input filters, see Creating Input Filters in AWS Elemental Delta.

Topics

- PUT Input Filter
- GET Input Filters List
- GET Input Filter
- DELETE Input Filter

PUT Input Filter

To modify the attributes of an input filter, send a PUT request for the filter. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/input_filters/id
```

where *id* is the unique ID of the `input_filter` to modify.

Body of HTTP

The body of your request is XML content consisting of only the tags to change. For tag details, see Creating Input Filters in AWS Elemental Delta.

Request Example

This request changes the `segmentation_marker` of the input filter with the ID 9. It changes the value to **`idr`**.

```
PUT http://10.24.34.2:8080/input_filters/9
-------------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<input_filters>
    <input_filter>
        <filter_settings>
            <segmentation_marker>idr</segmentation_marker>
        </filter_settings>
    </input_filter>
</input_filters>
```

Response

The response repeats back all the data in the input filter.

GET Input Filters List

To retrieve a list of input filters, send a GET request for the input filters. The following sections describe how to format the request.

HTTP URL

```
GET http://Delta IP address:8080/input_filters
```

Request Example

```
GET http://10.24.34.2:8080/input_filters
```

Response

The response is XML content consisting of one `input_filters` container that holds:

- Zero or more `input_filter` elements, each with:
  - An HREF that specifies the unique ID of the input filter and the product and version installed on the node.
  - `input_filter` tags.
  - One `filter_settings` element.

This example response specifies two input filters, the first with ID 9, the second with ID 10. Within the first filter, the filter settings have ID 3; within the second, the filter settings have ID 7.

```
<?xml version="1.0" encoding="UTF-8"?>
<input_filters href="/input_filters" product="Delta" version="2.3.0.123456">
    <input_filter href="/input_filters/9" product="Delta" version="2.3.0.123456">
        <filter_type>udp_input</filter_type>
        <label>Filter 9</label>
        <filter_settings>
            <id>3</id>
            <template_id/>
            <content_window_type>keep_seconds</content_window_type>
            <seconds_to_keep>900</seconds_to_keep>
            <storage_location>/data/server/content/b2/</storage_location>
            <segmentation_marker>idr</segmentation_marker>
            <enable_fec_rx>false</enable_fec_rx>
            <udp_input>
                <id>7</id>
                <uri>udp://10.6.85.34:5021</uri>
                <igmp_source/>
            </udp_input>
        </filter_settings>
    </input_filter>
    <input_filter href="/input_filters/10" product="Delta" version="2.3.0.123456">
        <filter_type>mss_input</filter_type>
        <label>Filter 10</label>
        <filter_settings>
            <id>7</id>
            .
            .
            .
    </input_filter>
</input_filters>
```

GET Input Filter

To retrieve details about a specific input filter, send a GET request for the input filter. The following sections describe how to format the request.

HTTP URL

```
GET http://Delta IP address:8080/input_filters/id
```

where *id* is the unique ID of the `input_filter` that you're viewing details for.

Request Example

```
GET http://10.24.34.2:8080/input_filters/46
```

Response

The response is XML content consisting of one `input_filter`, containing the same tags as the response for Creating Input Filters in AWS Elemental Delta.

This response retrieves the attributes for the input filter with the ID 46.

This example response is a representation. Your results may vary.

```
<?xml version="1.0" encoding="UTF-8"?>
<input_filter href="/input_filters/46" product="Delta" version="2.3.0.123456">
    <id>46</id>
    <label>Filter 46</label>
    <filter_type>udp_input</filter_type>
    <filter_settings_id>5</filter_settings_id>
    <filter_settings>
        <id>5</id>
        <storage_location>/data/server/content/delta1/</storage_location>
        <template_id nil="true" />
        <content_window_type>keep_seconds</content_window_type>
        <seconds_to_keep>14400</seconds_to_keep>
        <segmentation_marker>idr</segmentation_marker>
        <enable_fec_rx>false</enable_fec_rx>
        <buffer_time>30</buffer_time>
        <aws_credential_id nil="true" />
        <storage_file_duration>0</storage_file_duration>
        <vod_catalog_output_location nil="true" />
```

```
        <vod_catalog_output_aws_credential_id nil="true" />
        <vod_output_template_alias nil="true" />
        <vod_url_prefix nil="true" />
        <time_source>ingest_time</time_source>
        <udp_inputs>
            <udp_input>
                <id>5</id>
                <uri>udp://10.6.86.123:5021</uri>
                <igmp_source nil="true" />
                <interface>eth0</interface>
                <program_number nil="true" />
                <udp_input_filter_id>5</udp_input_filter_id>
            </udp_input>
        </udp_inputs>
    </filter_settings>
</input_filter>
```

DELETE Input Filter

To remove an input filter from Delta, send a DELETE request for the input filter.

When you remove an input filter, Delta doesn't delete the content associated with the input filter. The content is still functional, but the input filter doesn't ingest new video.

Take steps to ensure that the upstream encode is no longer delivering the asset before deleting the input filter.

You are not required to delete the content entity in Delta. The exception to this is UDP input filters. For these, you must delete both the filter and the content entity.

You can retain the content entity, or you may want to delete it, as described in DELETE a Content Entity.

HTTP URL

```
DELETE http://Delta IP address:8080/input_filters/id
```

where *id* is the unique ID of the `input_filter` to delete.

Response

A 200 OK response indicates the delete was successful.

Working with Input Users

Input users are agents who are authorized to send content to WebDAV and (optionally) RTMP input filters. The following sections describe actions that you can take against input users.

> **Note**
>
> There is no PUT command for an input user. You can't modify an existing input user. If you need to make changes to an input user, you must create a new one and delete the one you wanted to modify.

Topics

- POST Input User
- GET Input Users List
- GET Input User
- DELETE Input User

POST Input User

To create an input user for authentication, send a POST request. The following sections describe how to format the request.

HTTP URL

```
POST http://Delta IP address:8080/input_users
```

Body of HTTP

The body of your request is XML content consisting of one `input_user` container and the tags described in the following table.

Tag	Type	Description
`username`	string	The name for the input user. The username is case-sens itive.

Tag	Type	Description
`password`	string	The password for the input user.

Request Example

This request adds an input user with the name **`ida.wells`**.

```
POST http://10.24.34.2:8080/input_users
---------------------------------------------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<input_user>
    <username>ida.wells</username>
    <password>Secret!123</password>
</input_user>
```

Response

The response repeats back the data that included in the POST request, with the addition of a unique ID that Delta assigns.

The response is identical to the response to a GET Input User.

GET Input Users List

To retrieve a list of input users, send a GET request for the input users. The following sections describe how to format the request.

HTTP URL

```
GET http://Delta IP address:8080/input_users
```

Response

The response is XML content consisting of one `input_users` container that holds zero or more `input_user` elements, each with the following:

- An HREF that specifies the unique ID of the user and the product and version installed on the node.

- The `id`, `username`, and `password` tags.

The first user in the list is always the built-in user, elemental. Other users are those you added.

This example response is a representation. Your results may vary.

```
GET http://10.24.34.2:8080/input_users
----------------------------------------
<input_users href="/input_users" product="Delta" version="2.3.0.123456">
    <input_user href="/input_users/1" product="Delta" version="2.3.0.123456">
        <id>1</id>
        <username>elemental</username>
        <password>delta</password>
    </input_user>
    <input_user href="/input_users/2" product="Delta" version="2.3.0.123456">
        <id>2</id>
        <username>elemental2</username>
        <password>delta2</password>
    </input_user>
</input_users>
```

GET Input User

To retrieve details about a specific input user, send a GET request for the input user. The following sections describe how to format the request.

HTTP URL

```
GET http://Delta IP address:8080/input_users/id
```

where *`id`* is the unique ID of the `input_user` that you're viewing the details of.

Request

```
GET http://10.24.34.2:8080/input_users/2
```

Response

The response is XML content consisting of one XML content consisting of one `input_user` element, containing the `id`, `username`, and `password` tags.

This example response is a representation. Your results may vary.

```
<?xml version="1.0" encoding="UTF-8"?>
<input_user href="/input_users/2" product="Delta" version="2.3.0.123456">
    <id>2</id>
    <username>elemental2</username>
    <password>delta2</password>
</input_user>
```

DELETE Input User

To remove an input user from Delta, send a DELETE request for the input user.

HTTP URL

```
DELETE http://10.24.34.2:8080/input_users/id
```

where *id* is the unique ID of the `input_user` to delete.

Response

A 200 OK response indicates the delete was successful.

Working with a Live or VOD Contents Entity

A contents entity in AWS Elemental Delta represents the content that Delta ingested.

A content entity is created automatically or manually, as shown in the table below.

Content Type	Direction	Type of Content Entity	When Created
HTTP Input	Push	Content	Automatically, when an asset is pushed to Delta.
MSS Input	Push	Content	Automatically, when an asset is pushed to Delta.

Content Type	Direction	Type of Content Entity	When Created
Remote Input	Pull	Remote input content	Manually. See POST Remote Input Content.
RTMP Input	Push	Content	Automatically, when an asset is pushed to Delta.
UDP Input	Push	Content	Automatically, when an asset is pushed to Delta.
Watch Folder Input	Pull	Content	Automatically, the first time Delta detects an asset in the watched folder.
WebDAV Input	Push	Content	Automatically, when an asset is pushed to Delta.
VOD Catalog Input	Varies	VOD Catalog Content	See Working with a VOD Catalog Contents Entity.

Content Entities and Output Filters

The content entity is associated with output filters. The filters encapsulate information about how to package the content. At the API level, the output filters are part of the content entity: the path to an output filter is via the content entity.

For example, here is the URI for a content entity:

```
/contents/5
```

And here is the URL for an output filter associated with that content entity:

```
/contents/5/filters/3
```

Content Aliases

Delta automatically generates aliases for contents entities when they are created. The alias provides a more efficient way of referencing a contents entity, rather than providing the system-generated numerical ID. The alias can be used interchangeably with the assigned ID.

Example

You have three Delta nodes, each of which has a contents entity for ChannelX. Each of those entities has a unique ID:

- Delta1 has Content ID 22
- Delta2 has Content ID 5
- Delta 3 has Content ID 12

Rather than refer to each content by its ID, you can assign the same alias to each (as long as they are on separate Delta nodes). When you provide an alias **`ChannelX`** to each contents entity, you can use it to reference all content entities.

Auto-Generated Naming Conventions

The contents alias is generated using the input filter name and content name (as populated by the manifest) and is based on the number of manifests that are generated from the input filter.

Input Filter to Content	Alias	Input Filter	Content	Example
1 input filter: 1 content	<Input_filter_name>	Channel News	x	Channel_News
1 input filter: multiple contents	<Input_filter_name>_<contents_name>	Sports Live	Football	Sports_Live_Football

Input Filter to Content	Alias	Input Filter	Content	Example
	If the input filter and content's names are the same, only the input filter name is used to generate the alias. The name isn't duplicated in the alias.			

Aliases can include Unicode characters in addition to period (.), dash (-), and underscore (_). Note that:

- Invalid characters in the input filter or content name are translated to underscores (_) in the alias.
- The alias is NULL if:
  - The generated alias is longer than 255 characters.
  - The alias is an integer.
  - The alias is a duplicate.

Manual Alias Updates

The alias can be manually modified with a PUT request but must not:

- Include spaces, tabs, new lines or: ! @ # $ % ^ & * ( ) + = [ ] { } \ | ; : ‘ “ , < . > ? /
- Be an integer.
- Duplicate an existing alias.

The alias is generated only when you create the content. It is never automatically updated. This includes upgrades, in-flight changes to the input filter name, and content re-ingest. To automatically generate a new alias, you must create a new input filter.

Topics

- POST Remote Input Content
- PUT Remote Input Content
- GET Remote Content
- GET Remote Content for an Input Filter
- GET Contents List
- GET a Content and Filters
- GET a Content and Tracks
- PUT Content
- DELETE a Content Entity

POST Remote Input Content

To create a remote input content entity, send a POST request. The more input content entity is the only content type that has a POST request. There is no POST for other content entities because Delta creates the entities automatically when you create the corresponding input filter.

You can create more than one remote input content for the same remote input filter. For example, you might have one remote input content for each individual asset at the same remote location.

For general information about the options for creating input filters, contents, and filters, see Working with Input Filters.

HTTP URL

```
POST http://Delta IP address:8080/input_filters/filter_id/remote_input_contents
```

where the *`filter ID`* must be the ID of a remote input filter.

Body of HTTP

The body of your request is XML content consisting of one `remote_input_content` element that contains the tags in the following table.

Tag	Type	Description
name	string	A name for this Content.

Tag	Type	Description

`uri`	string	The final portion of the path to the remote content. This value is appended to the `remote_uri` of the associated Remote Input Filter.
		
		For example, the `remote_uri` might be:
		
		`http:/10.24.34.2/popular/interviews`
		
		The uri might be:
		
		`mendis.m3u8`
		
		Note that this URI tag is part of the body of a POST, but it does not get returned in a GET. Instead, the `uri` and `remote_uri` are shown in the path tag that is included in a GET.

Request Example

```
POST http://10.24.34.2:8080/input_filters/19/remote_input_contents
--------------------------------------------------------------------
<content>
    <name>Football Game 1</name>
    <uri>football_1.m3u8</uri>
</content>
```

Response

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

- `id`: A unique ID for the remote input content.
- `encrypted`: True if Delta has detected that the asset is encrypted.
- `path`: The path to the content, consisting of the `remote_uri` (from the remote input filter) and `uri` (from the content).
- `type`: Always HLS for a remote input content.
- `remote_input`: True for remote input content.

The response is identical to the response to a GET Content.

```
<?xml version="1.0" encoding="UTF-8"?>
<content href="/contents/13" product="Delta" version="2.3.0.123456">
    <id>13</id>
    <name>Football Game 1</name>
    <encrypted>false</encrypted>
    <path>http://10.1.1.1/out/u/football_1.m3u8</path>
    <type>HLS</type>
    <remote_input>true</remote_input>
</content>
```

PUT Remote Input Content

To modify the attributes of a remote input content entity, send a PUT request.

This command lets you specify an alias for the remote input content. You can only specify an alias via PUT, not POST.

HTTP URL

```
PUT http://Delta IP address:8080/input_filters/filter_id/remote_input_contents/id
```

where the *`filter id`* must be the ID of a remote input filter and *`id`* is the ID of the remote input content entity.

Body of HTTP

The body of your request is XML content consisting of one `remote_input_content` element that contains the tags in the following table.

Tag	Type	Description
`alias`	string	A name that you assign to the remote input content entity. Creating this name lets you reference the content by a customized name instead of by its ID tag. Must be unique and must contain at least one non-numeral. Valid characters are letters, numerals, underscor e, and dash.
`name`	string	A name for this content entity.

`uri`	string	The final portion of the path to the remote content. This value is appended to the `remote_uri` of the associated Remote Input Filter.
		
		For example, the `remote_uri` might be:
		
		`http:/10.24.34.2/p opular/interviews`
		
		The uri might be:
		
		`mendis.m3u8`
		
		Note that this URI tag is part of the body of a POST, but

Tag	Type	Description
		it does not get returned in a GET. Instead, the `uri` and `remote_uri` are shown in the path tag that is included in a GET.

Response

The response is identical to the response for POST Remote Input Content.

GET Remote Content

To get the attributes of a specified remote content, including its associated filters, send a GET request as described in GET a Content and Filters and specify the ID of the desired remote content.

GET Remote Content for an Input Filter

To retrieve the attributes of a remote input content entity that is associated with a specific input filter, send a GET request. The response includes the content entity and does not include any filters.

HTTP URL

```
GET http://Delta IP address:8080/input_filters/id/remote_input_contents
```

where the *id* is the unique ID of the input filter.

Response

The response is XML content consisting of one `contents` container that holds one or more `content` elements, one for each remote content associated with the specified input filter. Each content element contains:

- An HREF that specifies the path to this filter and the product and version installed on the node.
- Several tags as described in the following table.

Tag	Type	Description
`id`	integer	A unique ID for this content entity. Same as the ID in the HREF.
`name`	string	The value depends on the type of content entity: • HLS: Name of .m3u8 file for the asset. • MSS: Name of the .ism file for the asset. • MP4: Name of the .smil file for the asset. • TS: Name of the input filter associated with this content entity.
`alias`	string	The assigned Content alias.
`encrypted`	boolean	Read-only. `true` if Delta has detected that the asset is encrypted.
`path`	string	Only for these input filters: Remote Input, HTTP Put, WebDAV, and Watch Folder. The path to the content, as follows: • For HLS content from a remote input filter: `remote_uri` (from the filter) + `uri` (from the content entity)

Tag	Type	Description
		• For HLS content from a WebDAV or HTTP Put filter: `storage_location` (from the filter) + `name` (from the content entity) • For HLS or MP4 content from a Watch Folder filter: `incoming_uri` (from the filter) + `name` (from the content entity)
`status`	string	Always `remote_input` .
`storage_location`	string	Only for these input filters: Watch Folder and UDP. Identical to the value in the `storage_location` of the input filter associated with this content entity.
`remote_input`	boolean	`true` if the content is associated with a remote input filter. Otherwise `false`.
`type`	string	Specifies the content type: HLS, MSS, MP4, or TS. The type is controlled by the input filter, see Ingesting Content.

Tag	Type	Description
`vod`	boolean	• `true` if the content is being ingested as VOD • `false` if it is being ingested as Live content
`output_template`	integer	The ID of the output template. • If some or all of the filters in this content entity were created via an output template (by specifying a `template_id` in the `input_filter`), then this tag specifies the ID of that template. The filters themselves show in the filter array. • If some or all of the filters in this content were created by adding a `template_id` tag to this content entity after creation (by adding a template_id), then this tag specifies the ID of that template. The filters themselves show in the filter array. • If none of the filters in this content were created via an output template, this tag is null or omitted.

This request gets all the remote content associated with the input filter that has the ID 9.

```
GET http://10.24.34.2:8080/input_filters/9/remote_input_contents/
--------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<contents>
    <content href="/contents/13" product="Delta" version="2.3.0.123456">
        <id>13</id>
        <name>Football Game 1</name>
        <alias/>
        <encrypted>false</encrypted>
        <path>http:/10.1.1.1/out/u/football_1.m3u8</path>
        <status>remote_input</status>
        <storage_location/>
        <type>hls</type>
        <remote_input>true</remote_input>
        <vod>false</vod>
        <filter href="/contents/9/filters/21" product="Delta"
 version="2.3.0.123456">
            <id>21</id>
            .
            .
            .
        </filter>
    </content>
    <content href="/contents/22" product="Delta" version="2.3.0.123456">
        <id>22</id>
    </content>
<contents>
```

GET Contents List

To retrieve a list of all the content entities on the Delta node, along with some of the attributes of each content entity, send a GET request. The response does not include any filters.

This command works for all content entities (including remote input content entities) except for VOD Catalog content entities. See GET VOD Catalog Contents List.

HTTP URL

```
GET http://Delta IP address:8080/contents
```

Or

```
GET http://Delta IP address:8080/contents?page=value&per_page=value
```

Or

```
GET http://Delta IP address:8080/contents?summary=type[]=value&name=value
```

Add parameters to the GET request to filter the contents entities returned in the response. Use the `summary` parameter to enter filter criteria. See the following table for available filter options.

Response Filter	Valid Value	Example
name	All or part of a contents name. Can be an integer or string.	**`...summary?name=del`** Returns all contents entities with *del* anywhere in the name.
alias	All or part of a contents alias. Can be an integer or string.	**`...summary?alias=spo`** Returns all contents entities with *spo* anywhere in the alias.
id	A single integer.	**`...summary?id=16`** Returns the contents entity with an ID of 16.
type	You must enter brackets (`[]`) with no spaces after the filter name, then one of the following: • **`hls`** • **`ts`** • **`mss`**	**`...summary?type[]=hls`** Returns all HLS contents entities.

Response Filter	Valid Value	Example
	• **mp4** • **rtmp**	
status	You must enter brackets ([ ]) with no spaces after the filter name, then one of the following: • **active** • **stale** • **inactive** • **complete** • **remote_input**	**...summary?status[ ]=complete** Returns all contents entities in a complete status.

Enter multiple filters by separating them with an ampersand (&) and no spaces (as shown above in HTTP URL).

page and per_page

You can use page and per_page optional filter parameters to narrow the results that Delta includes in the response to your GET request.

- The per_page parameter chunks results into groups or *pages* of a given count, with the newest entries on page 1. For example, **30** means chunk entries into one page for IDs 1-30, another for 31-60, and so on.
- The page parameter indicates that you want results for the specified page (default is page 1). For example, if per_page is **30**, then page 3 contains the entries that appear in positions 61 to 90. If page is **3**, then you will see entries that are in positions 61 to 90.

You can use either one or both of the filter parameters in your GET request.

Request

```
GET  http://10.24.34.2:8080/contents
```

Response

The response is XML content consisting of one `contents` container that holds zero or more `content` elements, each with:

- An HREF that specifies the unique ID of the content and the product and version installed on the node.
- Several tags, as described in [The `content` Element](#).

```
<?xml version="1.0" encoding="UTF-8"?>
<contents href="/contents" product="Delta" version="2.3.0.123456">
    <content href="/contents/361" product="Delta" version="2.3.0.123456">
        <id>361</id>
        <type>HLS</type>
        <name>HLSencryptlive</name>
        <alias>hls_ingest_HLSencryptlive</alias>
        <path>/data/server/content/hlsingest/sigma/HLSencryptlive.m3u8</path>
        <bandwidth>0.00 kbit</bandwidth>
        <duration nil="true" />
        <finished>false</finished>
        <fragment_duration>12.012</fragment_duration>
        <total_file_size>262.60 MB</total_file_size>
        <total_duration>300.3</total_duration>
        <total_filters>1</total_filters>
        <total_endpoints>1</total_endpoints>
        <vod_content>false</vod_content>
        <remote_input>false</remote_input>
        <output_template_id nil="true" />
        <storage_location nil="true" />
        <uri>ingest/HLSencryptlive.m3u8</uri>
        <last_segment_time>2017-04-10T09:42:53-07:00</last_segment_time>
        <deleting>false</deleting>
        <encrypted>true</encrypted>
        <status>active</status>
        <vod>false</vod>
    </content>
    <content href="/contents/232" product="Delta" version="2.3.0.123456">
        .
        .
    </content>
</contents>
```

Topics

- The content Element

The content Element

The following tags are returned in response to a GET Content request.

Tag	Type	Description
`id`	integer	A unique ID for this contents entity. Same as the ID in the HREF.
`type`	string	This tag indicates the content type: HLS, MSS, MP4, RTMP, or TS. The type is controlle d by the input filter, see Ingesting Content.
`name`	string	The value depends on the type of content entity: • HLS: Name of .m3u8 file for the asset. • MSS: Name of the .ism file for the asset. • MP4: Name of the .smil file for the asset. • RTMP: Name of the asset. • TS: Name of the input filter associated with this contents entity.
`alias`	string	The assigned contents alias.

Tag	Type	Description
`path`	string	Only for these input filters: Remote Input, HTTP Put, WebDAV, and Watch Folder. The path to the content, as follows: • For HLS content from a remote input filter: `remote_uri` (from the filter) + `uri` (from the content entity) • For HLS content from a WebDAV or HTTP Put filter: `storage_location` (from the filter) + `name` (from the content entity) • For HLS or MP4 content from a Watch Folder filter: `incoming_uri` (from the filter) + `name` (from the content entity)
`bandwidth`	string	Read-only. Provides the total bitrate for all streams in this contents.
`duration`	string	Read-only.
`finished`	boolean	Read-only. If `true`, contents is in a completed status.
`fragment_duration`	string	Read-only.

Tag	Type	Description
`total_file_size`	string	Read-only. This string provides the total size of all stream sizes in this contents.
`total_duration`	string	Readonly. This string provides the total duration of all stream durations in this contents.
`total_filters`	integer	Read-only. This string provides the total number of output filters applied to this contents.
`total_endpoints`	integer	Read-only. This string provides the number of output filters that have endpoints enabled on this contents.
`vod_content`	boolean	Read-only. • `true` if the content is being ingested as VOD • `false` if it is being ingested as live content
`remote_input`	boolean	`true` if the content is associated with a remote input filter. Otherwise `false`.

Tag	Type	Description
`output_template_id`	integer	Read-only. When present, indicates the output template applied as specified on the input filter.
`storage_location`	string	Only for these input filters: Watch Folder and UDP. Identical to the value in the `storage_location` of the input filter associated with this content entity.
`uri`	string	Read-only. Content URI as defined on the input filter. For more information, see Creating Input Filters in AWS Elemental Delta.
`last_segment_time`	string	Read-only. Most recent segment received from any stream on this contents entity.
`deleting`	boolean	Read-only. When `true`, indicates that the contents is in the process of being deleted.

Tag	Type	Description
`encrypted`	boolean	Read-only. `true` if Delta has detected that the asset is encrypted. Only HLS and MSS content is capable of being encrypted content.
`status`	string	Indicates the type of content. • `true` for remote input content. • `false` for all other types of content.
`vod`	boolean	Read-only. If true, indicates VOD content.

GET a Content and Filters

To retrieve the attributes of a contents entity and all of its attributes, including its associated output fitlers, send a GET request. The response returns full details about the contents entity.

This command works for any content entity, including a remote content entity.

HTTP URL

```
GET http://Delta IP address:8080/contents/id
```

where the *id* is the unique ID or alias of the content.

Request

```
GET http://10.24.34.2:8080/contents/357
```

Response

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

- An HREF that specifies the path to this filter and the product and version installed on the node.
- Several tags, as described in The `content` Element.
- Zero or more `filter` elements, one for each output filter associated with the content. For information about the `filter` element, see Outputting Content from AWS Elemental Delta.

This example request gets the attributes for the content with the ID 357.

```
<?xml version="1.0" encoding="UTF-8"?>
<content href="/contents/357" product="Delta" version="2.3.0.123456">
    <id>357</id>
    <type>HLS</type>
    <name>HLSencrypt</name>
    <alias>hls_ingest_HLSencrypt</alias>
    <path>/data/server/content/hlsingest/sigma/HLSencrypt.m3u8</path>
    <bandwidth>0.00 kbit</bandwidth>
    <duration nil="true" />
    <finished>true</finished>
    <fragment_duration>12.012</fragment_duration>
    <total_file_size>256.99 MB</total_file_size>
    <total_duration>293.86</total_duration>
    <total_filters>1</total_filters>
    <total_endpoints>1</total_endpoints>
    <vod_content>true</vod_content>
    <remote_input>false</remote_input>
    <output_template_id nil="true" />
    <storage_location nil="true" />
    <uri>ingest/HLSencrypt.m3u8</uri>
    <last_segment_time>2017-04-10T09:34:13-07:00</last_segment_time>
    <deleting>false</deleting>
    <encrypted>true</encrypted>
    <status>complete</status>
    <vod>true</vod>
    <filters>
        <filter>
        .
        .
        .
            <filter_settings>
```

```
            .
            .
            </filter_settings>
        </filter>
    </filters>
</content>
```

GET a Content and Tracks

To retrieve the attributes of a contents entity and all of its audio, video, and captions tracks, send a GET request. The response returns full details about the contents entity.

This command works for any content entity, including a remote content entity.

HTTP URL

```
GET http://Delta IP address:8080/contents/id/tracks
```

where the *id* is the unique ID or alias of the content.

Request

```
GET http://10.24.34.2:8080/contents/9/tracks
```

Response

XML content consisting of one `tracks` container that holds zero or more `track` elements with:

- An HREF that specifies the path to this track and the product and version installed on the node.
- Zero or more `tracks` elements, one for each output audio, video, or captions track associated with the content.

  For information about the tracks tags, see Working with Tracks: GET Tracks.

This example request gets the tracks for the content with the ID 307.

```
<?xml version="1.0" encoding="UTF-8"?>
<tracks href="/contents/307/tracks" product="Delta" version="2.3.0.123456">
    <track href="/contents/307/tracks/415" product="Delta" version="2.3.0.123456">
        <id>415</id>
        <track_type>audio</track_type>
```

```
        <pid>2</pid>
        <codec>AACH</codec>
        <bitrate>65536</bitrate>
        <stream_index>1</stream_index>
        <language />
        <channel_count>2</channel_count>
        <sample_rate>44100</sample_rate>
        <sample_rate_denominator>2</sample_rate_denominator>
        <bits_per_sample>16</bits_per_sample>
    </track>
    <track href="/contents/307/tracks/414" product="Delta" version="2.3.0.123456">
        <id>414</id>
        <track_type>video</track_type>
        <pid>1</pid>
        <codec>H264</codec>
        <bitrate>1800000</bitrate>
        <stream_index>1</stream_index>
        <framerate_numerator>24</framerate_numerator>
        <framerate_denominator>1</framerate_denominator>
        <width>960</width>
        <height>540</height>
    </track>
    <track href="/contents/307/tracks/417" product="Delta" version="2.3.0.123456">
        <id>417</id>
        <track_type>video</track_type>
        <pid>1</pid>
        <codec>H264</codec>
        <bitrate>1800000</bitrate>
        <stream_index>1</stream_index>
        <framerate_numerator>24</framerate_numerator>
        <framerate_denominator>1</framerate_denominator>
        <width>960</width>
        <height>540</height>
    </track>
</tracks>
```

PUT Content

To make changes to a contents entity, send a PUT request. On live and VOD content, you can use PUT to:

- Modify the name. See the example below.

- Change an `alias`. Note that, when the contents entity is initially created, the alias is auto-generated.
- Add a filter. You can add a filter to a contents entity that currently has no filters or you can add an additional filter. See Methods for Creating Output Filters for details and examples.

> **Note**
>
> You cannot use PUT Content to modify or delete a filter that already exists in the contents entity. Instead, use Modifying Output Filters or Deleting an Output Filter: DELETE.

HTTP URL

```
PUT http://Delta IP address:8080/contents/id
```

where *`id`* is the unique ID or alias of the content entity to modify.

Body of HTTP

The body of your request is XML content consisting of one `content` element that contains one or both of the tags in the following table.

Tag	Type	Description
`name`	string	A new name for the contents entity.
`alias`	string	A name that you assign to the Content. Creating this name lets you reference the content alias instead of referencing the content by its ID tag. This name must be unique and contain at least one non-numeral. Valid character

Tag	Type	Description
		s are letters, numerals, underscore, and dash.

Request Example

This request adds an alias to the content with the ID 29.

```
PUT http://10.24.34.2:8080/contents/29
-----------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<content>
    <alias>ad_crazy_cat29</alias>
</content>
```

Response

The response returns all the attributes of the contents entity, including all of its filters.

DELETE a Content Entity

To remove a contents entity from Delta, send a DELETE request. Deleting the content also deletes all output `filters`, `filter_settings`, `stream_sets`, endpoints, and statistics. It also deletes the stored content. However, it does not delete the associated input filter.

There are two general cases for deleting a contents entity:

- You want to delete both the contents entity and its associated input filter because you no longer want to ingest the asset.

  In this case, you should take steps so that the upstream encoder is no longer delivering the asset. Then delete the input filter first, then delete the contents entity.

> **Note**
>
> In some cases, there could be unexpected results if you don't stop the upstream encoder before you delete the input filter and contents entity.

- You want to delete the contents entity but retain the input filter to re-activate the input filter at a later date to start ingesting the asset again. However, you do not want to retain the current stored content.

  In this case, you should take steps so that the upstream encoder is no longer delivering the asset. Then delete the contents entity.

  > **Note**
  > With a UDP input filter, you can't delete the contents entity but retain the input filter.

HTTP URL

```
DELETE http://Delta IP address:8080/contents/id
```

where *id* is the unique ID or alias of the contents entity to delete.

Response

A 200 OK response indicates the delete was successful.

Working with a VOD Catalog Contents Entity

A VOD Catalog contents entity is created from one of four paths listed in the following table.

> **Note**
> A VOD Catalog output template must already be defined to prior to initiating any of these paths. For information about creating output templates, see Working with Output Templates in AWS Elemental Delta.

Path	VOD Catalog Content Identified	When Created
Live-to-VOD Catalog Promotion	Live to VOD Catalog output filter is applied to a live contents entity. See The Live	Automatically created at the expiration time specified in the output filter.

Path	VOD Catalog Content Identified	When Created
	to VOD Catalog `filter_settings` Element for more information about output filters.	
VOD-to-VOD Catalog Promotion	VOD Catalog fields are completed on the input filter. See input filter settings in Creating Input Filters in AWS Elemental Delta for more information about input filters.	Automatically created when ingest of the VOD content is complete.
Existing VOD Content Promotion	The API is used to promote existing VOD content to VOD Catalog. See POST VOD Catalog Content for more information.	Automatically created when the POST API call is sent.
VOD File Ingest	The VOD Catalog Input filter ingests content when Delta receives the ingest command (either through the API or in the web interface for the input filter). See POST VOD Catalog - Initiate Ingest for more information about this call.	Automatically created when Delta ingests an existing VOD asset from an external source.

For more detailed information about VOD Catalog, see Working with VOD Catalog Assets in AWS Elemental Delta.

The following sections describe actions you can take against a VOD Catalog Content.

Topics

- POST VOD Catalog - Initiate Ingest
- POST VOD Catalog Content
- DELETE a Content Metadata
- PUT VOD Catalog Contents
- GET VOD Catalog Contents List
- GET a VOD Catalog Content
- DELETE a VOD Catalog Content Entity

POST VOD Catalog - Initiate Ingest

To initiate ingest of existing VOD content from an external source, send a POST request. The following sections describe how to format the request.

A VOD Catalog output template must already be defined to perform these steps. For information about creating output templates, see Working with Output Templates in AWS Elemental Delta.

HTTP URL

```
 POST http://Delta IP address:8080/input_filters/input_filter_id/
vod_catalog_contents
```

Body of HTTP

The body of your request is XML content consisting of one `vod_catalog_content` element that contains the tag described in the following table.

Tag	Type	Description
`relative_path`	string	The relative path to the where the VOD asset is stored externally.

Request Example

```
POST http://10.24.34.2:8080/input_filters/49/vod_catalog_contents
```

```
-------------------------------------------------------------------
<vod_catalog_content>
    <relative_path>path/to/asset.m3u8</relative_path
</vod_catalog_content>
```

Response

The response repeats back the data that you posted with the addition of the `vod_catalog_content_path`.

```
POST http://10.24.34.2:8080/input_filters/49/vod_catalog_contents
------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<vod_catalog_content href="input_filters/49/vod_catalog" product="Delta"
 version="2.3.0.123456">
    <vod_catalog>
        <vod_catalog_content_path>VOD_IF/vod_68</vod_catalog_content_path>
    </vod_catalog>
</vod_catalog_content>
```

POST VOD Catalog Content

To promote an existing VOD asset in Delta to VOD Catalog, send a POST request. The following sections describe how to format the request.

> **Note**
>
> A VOD Catalog output template must already be defined to perform these steps. For information about creating output templates, see Working with Output Templates in AWS Elemental Delta.

HTTP URL

```
POST http://Delta IP address:8080/contents/content_id/vod_catalog_contents
```

where *`content_id`* must be the ID or alias of an existing VOD contents entity.

Body of HTTP

The body of your request is XML content consisting of one `vod_catalog_content` element that contains the tags in the following table.

Tag	Type	Description
`vod_output_template_alias`	string	The alias of a VOD Catalog Output template. This is required on a VOD Catalog Input Filter and when VOD content is being converted to VOD Catalog after ingest. See Working with Output Templates in AWS Elemental Delta for more information.
`content_alias`	string	An optional string you can define to take place of the `vod_catalog_content_path` in the playback URL. The string must be unique or promotion to VOD Catalog Content will fail. The `content_alias` is not displayed in the response to the POST `vod_catalog_contents` request. To check the alias, submit a GET request on the `vod_catalog_content_path` as described in GET a VOD Catalog Content.

Request Example

```
POST http://10.24.34.2:8080/contents/4/vod_catalog_contents
-------------------------------------------------------------------
<vod_catalog_content>
    <vod_output_template_alias>Promotion</vod_output_template_alias>
    <content_alias>VODContentAlias</content_alias>
</vod_catalog_content>
```

Response

The response repeats back the data that you posted with the addition of the `vod_catalog_content_path`. The response doesn't include `content_alias`. The alias is applied once the VOD Catalog promotion is complete so won't show in any API responses until post-promotion.

```
POST http://10.24.34.2:8080/contents/4/vod_catalog_contents
-------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<vod_catalog_content href="/contents/4/vod_catalog_content" product="Delta"
 version="2.3.0.123456">
    <vod_catalog>
        <vod_catalog_content_path>WebDAV_VOD_cat_me/content_4</
vod_catalog_content_path>
    </vod_catalog>
</vod_catalog_content>
```

DELETE a Content Metadata

If you promoted Delta VOD content to VOD Catalog (as described in POST VOD Catalog Content), delete the specified VOD Content entity's metadata. This leaves the content untouched but deletes the database rows.

See Working with VOD Catalog Assets in AWS Elemental Delta for more information.

HTTP URL

```
DELETE http://Delta IP address:8080/contents/id/metadata
```

Response

A 200 OK response indicates the delete was successful.

PUT VOD Catalog Contents

To make changes to a VOD Catalog contents entity, send a PUT request. On VOD Catalog content, you can change the VOD Catalog output template or assign a `name`, `resourceid`, or `content_alias`.

HTTP URL for VOD Catalog Contents

```
PUT http://Delta IP address:8080/vod_catalog_contents/id
```

where *`id`* is the unique path of the content.

HTTP URL with Content Alias

```
PUT http://Delta IP address:8080/vod_catalog_contents/valias/content alias
```

where *`content alias`* is the alias that you defined in the Live to VOD Catalog output filter or in the VOD content promotion request.

Body of HTTP

The body of your request is XML content consisting of one `vod_catalog_content` element that contains the tags in the following table.

Tag	Type	Description
`vod_output_template_alias`	string	An alias for a different VOD Catalog output template.
`name`	string	A unique name for the VOD Catalog Content. When using replacement tokens, the `name` value replaces all instances of $name$ on output filters at egress.

Tag	Type	Description
`content_alias`	string	An optional string you can define to take place of the `vod_catalog_content_path` in the playback URL. The string must be unique.
`resource_id`	string	The `resource_id` value replaces all instances of $resourceid$ on Encryption Output Filters at egress. This allows you flexibility in the output template so that you can provide a unique value for DRM and encryption settings on each endpoint. For more information, see *Output Template Replacement Tokens* in the AWS Elemental Delta 2.3 User Guide.

Tag	Type	Description
`resource_id_alt`	string	The `resource_id_alt` value replaces all instances of $resourceid_alt$ on Encryption Output Filters at egress. It works the same way as `resource_id` (above) and acts as an additional unique value on Encryption Output Filters. For more information, see *Output Template Replacement Tokens* in the AWS Elemental Delta 2.3 User Guide.

Request Example

This request changes the output template alias used to **`alternate`**.

```
PUT http://10.24.34.2:8080/vod_catalog_contents/WebDAV_VOD/content_4
------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<vod_catalog_content href="/vod_catalog_contents/WebDAV_VOD/content_4"
 product="Delta" version="2.3.0.123456">
    <vod_output_template_alias>alternate</vod_output_template_alias>
</vod_catalog_content>
```

Response

The response returns all the attributes of the content entity.

GET VOD Catalog Contents List

To retrieve a list of all the VOD Catalog contents entities on the Delta node, send a GET request. The response doesn't include any filters, but does have some of the attributes of each contents entity.

HTTP URL for VOD Catalog Contents

```
GET http://Delta IP address:8080/vod_catalog_contents
```

page and per_page

You can use page and per_page optional filter parameters to narrow the results that Delta includes in the response to your GET request.

- The per_page parameter chunks results into groups or *pages* of a given count, with the newest entries on page 1. For example, **30** means chunk entries into one page for IDs 1-30, another for 31-60, and so on.
- The page parameter indicates that you want results for the specified page (default is page 1). For example, if per_page is **30**, then page 3 contains the entries that appear in positions 61 to 90. If page is **3**, then you will see entries that are in positions 61 to 90.

You can use either one or both of the filter parameters in your GET request.

Response

The response is XML content consisting of one vod_catalog_contents container that holds zero or more vod_catalog_content elements, each with:

- An HREF that specifies the unique ID of the content and the product and version installed on the node.
- Several tags, as described in The vod_catalog_content Element.

The following example is a representation. Your results may vary.

```
GET http://10.24.34.2:8080/vod_catalog_contents
------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<vod_catalog_contents>
    <vod_catalog_content>
        <name></name>
```

```
        <path>default_hlscaptions/vod_32</path>
        <content_alias nil="true"/>
        <resource_id></resource_id>
        <resource_id_alt></resource_id_alt>
        <delta_owned>true</delta_owned>
        <vod_output_template_alias>vod</vod_output_template_alias>
        <filters>
            <filter>
            .
            .
                <filter_settings>
                .
                .
                </filter_settings>
                <stream_sets/>
            </filter>
        </filters>
    </vod_catalog_content>
</vod_catalog_contents>
```

Topics

- The vod_catalog_content Element

The `vod_catalog_content` Element

The following table describes the settings for the `vod_catalog_content` element.

Tag	Type	Description
`name`	string	A unique name for the VOD Catalog Content. When using replacement tokens, the `name` value replaces all instances of $name$ on output filters at egress.
`path`	string	Unique identifier for the VOD Catalog Content entity.

Tag	Type	Description
`content_alias`	string	String defined to take place of the `vod_catalog_content_path` in the playback URL.
`resource_id`	string	The `resource_id` value replaces all instances of $resourceid$ on Encryption Output Filters at egress. This allows you flexibility in the output template so that you can provide a unique value for DRM and encryption settings on each endpoint. For more information, see *Output Template Replacement Tokens* in the AWS Elemental Delta 2.3 User Guide.
`resource_id_alt`	string	The `resource_id_alt` value replaces all instances of $resourceid_alt$ on Encryption Output Filters at egress. It works the same way as `resource_id` (above) and acts as an additional unique value on Encryption Output Filters. </para> For more information, see *Output Template Replacement Tokens* in the AWS Elemental Delta 2.3 User Guide.

Tag	Type	Description
`delta_owned`	boolean	When **`true`**, Delta takes ownership of the content after ingest. This means that when the VOD Catalog Content is deleted from Delta, the original data is also deleted.
`vod_output_template_alias`	string	An alias for a VOD Catalog output template.
`filters`	object	Output filters applied from the VOD Catalog output template. For information about output filters, see Outputting Content from AWS Elemental Delta.

GET a VOD Catalog Content

To view all information about a VOD Catalog Content entity, send a GET request on the `path` provided in the GET `vod_catalog_contents` list response, like this:

```
GET http://Delta IP address:8080/vod_catalog_contents/content path
```

If the VOD Catalog content has a `content_alias`, the request URL is as follows:

```
GET http://Delta IP address:8080/vod_catalog_contents/valias/content path
```

DELETE a VOD Catalog Content Entity

To remove a VOD Catalog contents entity from Delta, send a DELETE request for the entity.

HTTP URL for VOD Catalog Contents

```
GET http://Delta IP address:8080/vod_catalog_contents/valias/content alias
```

where *id* is the unique path of the content.

HTTP URL with Content Alias

```
DELETE http://Delta IP address:8080/vod_catalog_contents/valias/content alias
```

where *content alias* is the alias that you defined in the Live to VOD Catalog output filter or in the VOD Content promotion request.

Response

A 200 response indicates the delete was successful.

Creating Input Filters in AWS Elemental Delta

The following sections describe how to create input filters. The type of input filter to create is determined by what type of content you're sending to Delta and how you send it. For information about the input filter types, see the AWS Elemental Delta 2.3 User Guide.

Input Filters and Output Templates

When you create an input filter, you can identify an output template in the `template_id` tag. Delta ingests content through the input filter and creates a contents entity. If you've identified an output template on the input filter, then Delta automatically applies output filters to the contents entity when it's created. If you don't identify an output template, then you have to manually add output filters.

Topics

- POST: Create an HTTP PUT Input Filter
- POST: Create an MSS Input Filter
- POST: Create a Remote Input Filter
- POST: Create an RTMP Input Filter
- POST: Create a UDP Input Filter
- POST: Create a Watch Folder Input Filter
- POST: Create a WebDAV Input Filter
- POST: Create a VOD Catalog Input Filter

POST: Create an HTTP PUT Input Filter

To create an HTTP PUT input filter and ingest content, send a POST request. The following sections describe how to format the request and provide examples of the request and response.

HTTP URL

```
POST http://Delta IP address:8080/input_filters
```

Body of HTTP

The body of your request is XML content consisting of one `input_filter` element that contains the following:

- General settings.
- One `filter_settings` element.

  See The HTTP PUT `filter_settings` Element for tag details.

> **Note**
>
> A VOD Catalog output template is required when you create a VOD Catalog input filter, use VOD Catalog elements in other input filters, or if you promote any content to VOD Catalog after it has been ingested. For information about output templates, see Working with Output Templates in AWS Elemental Delta.
> For information about VOD catalog, see Working with VOD Catalog Assets in AWS Elemental Delta.

Request Example

See the UDP Request Example for a general input filter request.

Response

The response repeats back the data that you posted, with the addition of unique IDs that Delta assigns to the `input_filter` and to the `filter_settings`.

The response is identical to the response to a GET Input Filter. SeeGET Input Filter for an example.

Topics

- The input_filter Element
- The HTTP PUT filter_settings Element

The `input_filter` Element

All input filters contain these tags. For convenience, they are repeated in each input filter section.

Tag	Type	Description
`filter_type`	string	One of these: • `http_put` • `mss_input` • `remote_input` • `rtmp_input` • `udp_input` • `watch_folder_input` • `webdav_input` • `vod_catalog_input`
`label`	string	Name of the filter. For a UDP input filter, Delta copies this value to the `name` tag of the content object associated with this input filter. If blank, Delta automatically generates the value as 'Filter *`[Delta-assigned ID number]`* '. For example, `Filter 8`.
`filter_settings`		A setting that holds tags applicable to the input filter types. For applicable tags by filter, see the following sections: • POST: Create an HTTP PUT Input Filter • POST: Create an MSS Input Filter • POST: Create a Remote Input Filter

Tag	Type	Description
		• POST: Create an RTMP Input Filter • POST: Create a UDP Input Filter • POST: Create a Watch Folder Input Filter • POST: Create a WebDAV Input Filter • POST: Create a VOD Catalog Input Filter

The HTTP PUT `filter_settings` Element

The following table describes the settings for an HTTP PUT input filter.

Tag	Type	Description
`automatically_promote_after_ingest`	boolean	For use only with VOD content. When **`true`**, Delta automatically converts VOD content to VOD catalog content upon completion.
`aws_credential_id`	integer	This field applies only if `storage_location` is an S3 URL. Enter the Amazon Web Service credential to use in order to be authenticated on the S3 server.
`content_window_type`	string	Specifies how long to store content:

Tag	Type	Description
		• `keep_seconds` : Enter the duration in `seconds_to_keep` . Delta saves content up to this limit. As new content is added, Delta removes old content on a first in, first out basis. If you plan to create a passthrough output filter for the content associated with this input filter, then make sure that this `content_window_type` is equal to or greater than the window of the upstream encoder/packager. Otherwise, playback can be disrupted. • `keep_all`: Never automatically discard content. This option is intended for short-lived content such as events or VOD content. When you're no longer using the content object and you delete it, Delta also deletes the stored content. This option is not recommended for linear sources because it eventually uses up all of your storage.

Tag	Type	Description
		• `packager_controlled` : WebDAV only. Content can be deleted by sending a DELETE request to the WebDAV server. The DELETE is sent from outside Delta, from your upstream packager that is posting to the WeDAV server on Delta. If you plan to create a live-to-VOD output filter for the content associated with this input filter, then the upstream encoder must be configured to retain content that is at least the same size as that intended VOD clip. For all values, keep in mind that Delta continues to ingest and store content, even if you later remove all output filters from this content object.
`encrypt_storage`	boolean	When **`true`**, AWS Elemental Delta encrypts ingested content on storage.
`full_url`	string	The primary node's absolute path using the node's hostname.

Tag	Type	Description
`promotion_delay_seconds`	integer	When `automatically_promote_after_ingest` is **`true`**, `promotion_delay_seconds` indicates how many seconds the system waits after initial ingest to convert VOD content to VOD catalog content.
`public_endpoint`	string	The primary node's absolute path using the node's IP address.

Tag	Type	Description
`relative_uri`	string	For HTTP or MSS: • If content has been posted to <Delta IP address>: 8080/in_mss/ or ../in_http/, then leave blank. • If it has been posted to a subfolder of `<Delta IP address>:8080/in_mss/` or `../in_http` ., then specify that subfolder path in format `<subfolder/sub-subfolder/>` (no leading slash). For WebDAV: • If content has been posted to `<WebDAV server>/in/` , leave blank. • If it has been posted to a subfolder of `<WebDAV server>/in/` , then specify that subfolder path in format `<subfolder/sub-subfolder/>` (no leading slash).
`seconds_to_keep`	integer	The number of seconds of content to keep on disk. Required if content_window_type is set to keep_seconds; ignored otherwise.

Tag	Type	Description

`storage_location`	string	The folder where Delta stores incoming content. Either:
		• A local directory that the Delta node has access to. Choose from one of the following folders or a folder under one of these. If the subfolder does not yet exist, Delta automatically creates it.
		  • `mnt`
		  • `tmp`
		  • `local_sources`
		  • `server`
		  • `pgsql`
		• A remote server: Any remote servers you mount via Settings. Note that you typically don't specify remote server storage for remote input.
		  If you have deployed a Delta cluster (for redundancy), any remote server you specify should be mounted on both nodes.
		• An Amazon Web Services S3 URL. Enter the URL as s3://<full path>

Tag	Type	Description
`template_id`	integer	Optional. Do one of the following: • Specify a template ID to associate with this input filter. Delta automatically associates the output filters belonging to the specified template with the content when the content object is created. • Omit this tag. You manually add output filters to the content after it has been created. See also Working with Output Templates in AWS Elemental Delta.
`time_source`	string	The time source used for Live to VOD clipping. Tags for the `time_source` element vary according to the input filter. For more information, see The time_source element.
`udp_input`	object	See The udp_input element.
`vips`	integer	The absolute paths for the virtual IP addresses (VIPs) of leader and secondary Delta nodes in a cluster.

Tag	Type	Description
`vod_content`	boolean	This field applies only to input filters that handle assets that could be either a stream or VOD. • If the asset is a stream, set the value to **`false`**. (A value of true will give unpredictable results.) • If the asset is VOD, set the value to **`true`**. (A value of false will give unpredictable results.) Default is `false`.
`vod_catalog_output_aws_credential_id`	integer	This field applies only if `vod_catalog_output_location` is an S3 URL. Enter the Amazon Web Service credential to use in order to be authenticated on the S3 server.

Tag	Type	Description

`vod_catalog_output_location`	string	Folder to store VOD Catalog contents produced using a Live-to-VOD Catalog output filter. Either:
		• A local directory that the Delta node has access to - one of the following folders or a folder under one of these (if the subfolder does not yet exist, it will be automatically created):
		  • `mnt`
		  • `tmp`
		  • `local_sources`
		  • `server`
		  • `pgsql-delta-vod`
		  • `pgsql`
		• An Amazon Web Services S3 URL. Enter the URL as `s3://<full path>`

`vod_output_template_alias`	string	The alias of a VOD Catalog output template. Required on a VOD Catalog input filter and when you're converting VOD content to VOD catalog after ingest. For information about creating a VOD Catalog output template, see Working with Output Templates in AWS Elemental Delta.

Tag	Type	Description
`vod_url_prefix`	string	Optional. Partial URL path added to the start of all VOD Catalog content endpoints produced through the input filter.

The `time_source` Element

The following table describes the settings for the `time_source` element.

Tag	Type	Description
`ingest_time`	string	Time from the Delta node.
`embedded`	string	Timecode for the source video. Times are assumed to be Coordinated Universal Time (UTC). If `embedded` is used and is not available in the stream, `ingest_time` is used instead.
`program_date_time`	string	Time from the external source. If `program_date_time` is used and is not available in the stream, `ingest_time` is used instead.

The `udp_input` Element

The following table describes the settings for the `udp_input` element.

Tag	Type	Description
`uri`	string	The URI of the UDP or RTP asset to ingest. Format: `<protocol>://<IP address>:<port>` Example: `udp://239.255.1.10:5001` If the firewall is enabled, make sure this port is on the list of open incoming ports (**Settings** > **Firewall**).
`igmp_source`	string	Optional. The IP address for source-specific multicast streams.
`interface`	string	Optional. Network interface to use (such as `eth2`). If not specified, Delta uses the system routing table to select an interface.
`program_number`	integer	For use with Multi-program Transport Streams (MPTS) to indicate which program's tracks Delta ingests.

POST: Create an MSS Input Filter

To create an MSS input filter and ingest content, send a `POST` request. The following sections describe how to format the request and provide examples of the request and response.

HTTP URL

```
POST http://Delta IP address:8080/input_filters
```

Body of HTTP

The body of your request is XML content consisting of one `input_filter` element that contains the following:

- General settings.
- One `filter_settings` element.

  See The MSS `filter_settings` Element for tag details.

> **Note**
>
> A VOD Catalog output template is required when you create a VOD Catalog input filter, use VOD Catalog elements in other input filters, or if you promote any content to VOD Catalog after it has been ingested. For information about output templates, see Working with Output Templates in AWS Elemental Delta.
> For information about VOD catalog, see Working with VOD Catalog Assets in AWS Elemental Delta.

Request Example

See the UDP Request Example for a general input filter request.

Response

The response repeats back the data that you posted, with the addition of unique IDs that Delta assigns to the `input_filter` and to the `filter_settings`.

The response is identical to the response to a GET Input Filter. SeeGET Input Filter for an example.

Topics

- The input_filter Element
- The MSS filter_settings Element

The `input_filter` Element

All input filters contain these tags. For convenience, they are repeated in each input filter section.

Tag	Type	Description
`filter_type`	string	One of these: • `http_put` • `mss_input` • `remote_input` • `rtmp_input` • `udp_input` • `watch_folder_input` • `webdav_input` • `vod_catalog_input`
`label`	string	Name of the filter. For a UDP input filter, Delta copies this value to the `name` tag of the content object associated with this input filter. If blank, Delta automatically generates the value as 'Filter *`[Delta-assigned ID number]`* '. For example, `Filter 8`.
`filter_settings`		A setting that holds tags applicable to the input filter types. For applicable tags by filter, see the following sections: • POST: Create an HTTP PUT Input Filter

Tag	Type	Description
		• POST: Create an MSS Input Filter • POST: Create a Remote Input Filter • POST: Create an RTMP Input Filter • POST: Create a UDP Input Filter • POST: Create a Watch Folder Input Filter • POST: Create a WebDAV Input Filter • POST: Create a VOD Catalog Input Filter

The MSS `filter_settings` Element

The following table describes the settings for an MSS input filter.

Tag	Type	Description
`automatically_promote_after_ingest`	boolean	For use only with VOD content. When **`true`**, Delta automatically converts VOD content to VOD catalog content upon completion.
`aws_credential_id`	integer	This field applies only if `storage_location` is an S3 URL. Enter the Amazon Web Service credential to use

Tag	Type	Description
		in order to be authenticated on the S3 server.

Tag	Type	Description
`content_window_type`	string	Specifies how long to store content: • `keep_seconds` : Enter the duration in `seconds_to_keep` . Delta saves content up to this limit. As new content is added, Delta removes old content on a first in, first out basis. If you plan to create a passthrough output filter for the content associated with this input filter, then make sure that this `content_window_type` is equal to or greater than the window of the upstream encoder/packager. Otherwise, playback can be disrupted. • `keep_all`: Never automatically discard content. This option is intended for short-lived content such as events or VOD content. When you're no longer using the content object and you delete it, Delta also deletes the stored content. This option is not recommended for

Tag	Type	Description
		linear sources because it eventually uses up all of your storage. • `packager_controlled` : WebDAV only. Content can be deleted by sending a DELETE request to the WebDAV server. The DELETE is sent from outside Delta, from your upstream packager that is posting to the WeDAV server on Delta. If you plan to create a live-to-VOD output filter for the content associated with this input filter, then the upstream encoder must be configured to retain content that is at least the same size as that intended VOD clip. For all values, keep in mind that Delta continues to ingest and store content, even if you later remove all output filters from this content object.
`encrypt_storage`	boolean	When **`true`**, AWS Elemental Delta encrypts ingested content on storage.

Tag	Type	Description
`full_url`	string	The primary node's absolute path using the node's hostname.
`promotion_delay_seconds`	integer	When `automatically_promote_after_ingest` is **`true`**, `promotion_delay_seconds` indicates how many seconds the system waits after initial ingest to convert VOD content to VOD catalog content.
`public_endpoint`	string	The primary node's absolute path using the node's IP address.

Tag	Type	Description
`relative_uri`	string	For HTTP or MSS: • If content has been posted to <Delta IP address>: 8080/in_mss/ or ../in_http/, then leave blank. • If it has been posted to a subfolder of `<Delta IP address>:8080/in_mss/` or `../in_http` ., then specify that subfolder path in format `<subfolder/sub-subfolder/>` (no leading slash). For WebDAV: • If content has been posted to `<WebDAV server>/in/` , leave blank. • If it has been posted to a subfolder of `<WebDAV server>/in/` , then specify that subfolder path in format `<subfolder/sub-subfolder/>` (no leading slash).
`seconds_to_keep`	integer	The number of seconds of content to keep on disk. Required if content_window_type is set to keep_seconds; ignored otherwise.

Tag	Type	Description

`storage_location`	string	The folder where Delta stores incoming content. Either:
		• A local directory that the Delta node has access to. Choose from one of the following folders or a folder under one of these. If the subfolder does not yet exist, Delta automatically creates it.
		  • `mnt`
		  • `tmp`
		  • `local_sources`
		  • `server`
		  • `pgsql`
		• A remote server: Any remote servers you mount via Settings. Note that you typically don't specify remote server storage for remote input.
		  If you have deployed a Delta cluster (for redundancy), any remote server you specify should be mounted on both nodes.
		• An Amazon Web Services S3 URL. Enter the URL as s3://<full path>

Tag	Type	Description
`template_id`	integer	Optional. Do one of the following: • Specify a template ID to associate with this input filter. Delta automatically associates the output filters belonging to the specified template with the content when the content object is created. • Omit this tag. You manually add output filters to the content after it has been created. See also Working with Output Templates in AWS Elemental Delta.
`time_source`	string	The time source used for Live to VOD clipping. Tags for the `time_source` element vary according to the input filter. For more information, see The time_source element.
`vips`	integer	The absolute paths for the virtual IP addresses (VIPs) of leader and secondary Delta nodes in a cluster.

Tag	Type	Description
`vod_catalog_output_aws_credential_id`	integer	This field applies only if `vod_catalog_output_location` is an S3 URL. Enter the Amazon Web Service credential to use in order to be authenticated on the S3 server.

`vod_catalog_output_location`	string	Folder to store VOD Catalog contents produced using a Live-to-VOD Catalog output filter. Either:
		• A local directory that the Delta node has access to - one of the following folders or a folder under one of these (if the subfolder does not yet exist, it will be automatically created):
		  • `mnt`
		  • `tmp`
		  • `local_sources`
		  • `server`
		  • `pgsql-delta-vod`
		  • `pgsql`
		• An Amazon Web Services S3 URL. Enter the URL as `s3://<full path>`

Tag	Type	Description
`vod_output_template_alias`	string	The alias of a VOD Catalog output template. Required on a VOD Catalog input filter and when you're converting VOD content to VOD catalog after ingest. For information about creating a VOD Catalog output template, see Working with Output Templates in AWS Elemental Delta.
`vod_url_prefix`	string	Optional. Partial URL path added to the start of all VOD Catalog content endpoints produced through the input filter.

The `time_source` Element

The following table describes the settings for the `time_source` element.

Tag	Type	Description
`ingest_time`	string	Time from the Delta node.

POST: Create a Remote Input Filter

To create a Remote Input filter and ingest content, send a POST request. The following sections describe how to format the request and provide examples of the request and response.

HTTP URL

```
POST http://Delta IP address:8080/input_filters
```

Body of HTTP

The body of your request is XML content consisting of one `input_filter` element that contains the following:

- General settings.
- One `filter_settings` element.

  See The Remote Input `filter_settings` Element for tag details.

> **Note**
>
> A VOD Catalog output template is required when you create a VOD Catalog input filter, use VOD Catalog elements in other input filters, or if you promote any content to VOD Catalog after it has been ingested. For information about output templates, see Working with Output Templates in AWS Elemental Delta.
> For information about VOD catalog, see Working with VOD Catalog Assets in AWS Elemental Delta.

Request Example

See the UDP Request Example for a general input filter request.

Response

The response repeats back the data that you posted, with the addition of unique IDs that Delta assigns to the `input_filter` and to the `filter_settings`.

The response is identical to the response to a GET Input Filter. SeeGET Input Filter for an example.

Topics

- The input_filter Element
- The Remote Input filter_settings Element

The `input_filter` Element

All input filters contain these tags. For convenience, they are repeated in each input filter section.

Tag	Type	Description
`filter_type`	string	One of these: • `http_put` • `mss_input` • `remote_input` • `rtmp_input` • `udp_input` • `watch_folder_input` • `webdav_input` • `vod_catalog_input`
`label`	string	Name of the filter. For a UDP input filter, Delta copies this value to the `name` tag of the content object associated with this input filter. If blank, Delta automatically generates the value as 'Filter *`[Delta-assigned ID number]`* '. For example, `Filter 8`.
`filter_settings`		A setting that holds tags applicable to the input filter types. For applicable tags by filter, see the following sections: • POST: Create an HTTP PUT Input Filter

Tag	Type	Description
		• POST: Create an MSS Input Filter • POST: Create a Remote Input Filter • POST: Create an RTMP Input Filter • POST: Create a UDP Input Filter • POST: Create a Watch Folder Input Filter • POST: Create a WebDAV Input Filter • POST: Create a VOD Catalog Input Filter

The Remote Input `filter_settings` Element

The following table describes the settings for a Remote Input filter.

Tag	Type	Description
`aws_credential_id`	integer	This field applies only if `storage_location` is an S3 URL. Enter the Amazon Web Service credential to use in order to be authenticated on the S3 server.
`bind_filters_to_output_template`	boolean	Applies only to remote input filters that have a value in the `template_id` tag. • `true` means that when the associated output template

Tag	Type	Description
		changes, the (possibly modified) output filters from the template are re-attached to the content associated with this filter. • `false` is the default. When the output template changes, the existing output filters are left unchanged.
`disk_cache_size`	integer	The size of the disk cache (in MB) for content associate d with this filter. Content saved up to this limit. As new content is added over the limit, old content is removed on a first in, first out basis.
`memory_cache_size`	integer	The size of the cache to memory (in MB) for content associated with this filter. We recommended that this value is less than or equal to `disk_cache_size` .

Tag	Type	Description

`remote_uri`	string	The base URI of the remote origin server that Delta fetches content from. Individual content paths consist of this remote_uri appended by uri (from the remote input content associated with this input filter). The following shows the format.
		`http://<server><path><file.m3u8>`
		For example, the remote_uri might be the following.
		`http:/10.24.34.2/popular/interviews`
		The uri might be the following.
		`2015_May/mendis.m3u8`

Tag	Type	Description

`storage_location`	string	The folder where Delta stores incoming content. Either:
		• A local directory that the Delta node has access to. Choose from one of the following folders or a folder under one of these. If the subfolder does not yet exist, Delta automatically creates it.
		  • `mnt`
		  • `tmp`
		  • `local_sources`
		  • `server`
		  • `pgsql`
		• A remote server: Any remote servers you mount via Settings. Note that you typically don't specify remote server storage for remote input.
		
		  If you have deployed a Delta cluster (for redundancy), any remote server you specify should be mounted on both nodes.
		• An Amazon Web Services S3 URL. Enter the URL as s3://<full path>

Tag	Type	Description
`template_id`	integer	Optional. Do one of the following: • Specify a template ID to associate with this input filter. Delta automatically associates the output filters belonging to the specified template with the content when the content object is created. • Omit this tag. You manually add output filters to the content after it has been created. See also Working with Output Templates in AWS Elemental Delta.
`udp_input`	object	See The udp_input element.

Tag	Type	Description
`vod_content`	boolean	This field applies only to input filters that handle assets that could be either a stream or VOD. • If the asset is a stream, set the value to **`false`**. (A value of true will give unpredictable results.) • If the asset is VOD, set the value to **`true`**. (A value of false will give unpredictable results.) Default is `false`.

The `udp_input` Element

The following table describes the settings for the `udp_input` element.

Tag	Type	Description
`uri`	string	The URI of the UDP or RTP asset to ingest. Format: `<protocol>://<IP address>:<port>` Example: `udp://239.255.1.10:5001` If the firewall is enabled, make sure this port is on the list of open incoming ports (**Settings** > **Firewall**).

Tag	Type	Description
`igmp_source`	string	Optional. The IP address for source-specific multicast streams.
`interface`	string	Optional. Network interface to use (such as `eth2`). If not specified, Delta uses the system routing table to select an interface.
`program_number`	integer	For use with Multi-program Transport Streams (MPTS) to indicate which program's tracks Delta ingests.

POST: Create an RTMP Input Filter

To create an RTMP input filter and ingest content, send a POST request. The following sections describe how to format the request and provide examples of the request and response.

HTTP URL

```
POST http://Delta IP address:8080/input_filters
```

Body of HTTP

The body of your request is XML content consisting of one `input_filter` element that contains the following:

- General settings.
- One `filter_settings` element.

  See [The RTMP `filter_settings` Element](#) for tag details.

> **Note**
>
> A VOD Catalog output template is required when you create a VOD Catalog input filter, use VOD Catalog elements in other input filters, or if you promote any content to VOD Catalog after it has been ingested. For information about output templates, see Working with Output Templates in AWS Elemental Delta.
> For information about VOD catalog, see Working with VOD Catalog Assets in AWS Elemental Delta.

Request Example

See the UDP Request Example for a general input filter request.

Response

The response repeats back the data that you posted, with the addition of unique IDs that Delta assigns to the `input_filter` and to the `filter_settings`.

The response is identical to the response to a GET Input Filter. SeeGET Input Filter for an example.

Topics

- The input_filter Element
- The RTMP filter_settings Element

The `input_filter` Element

All input filters contain these tags. For convenience, they are repeated in each input filter section.

Tag	Type	Description
`filter_type`	string	One of these: • `http_put` • `mss_input` • `remote_input`

Tag	Type	Description
		• `rtmp_input` • `udp_input` • `watch_folder_input` • `webdav_input` • `vod_catalog_input`
`label`	string	Name of the filter. For a UDP input filter, Delta copies this value to the `name` tag of the content object associated with this input filter. If blank, Delta automatically generates the value as 'Filter *`[Delta-assigned ID number]`* '. For example, `Filter 8`.

Tag	Type	Description
`filter_settings`		A setting that holds tags applicable to the input filter types. For applicable tags by filter, see the following sections: • POST: Create an HTTP PUT Input Filter • POST: Create an MSS Input Filter • POST: Create a Remote Input Filter • POST: Create an RTMP Input Filter • POST: Create a UDP Input Filter • POST: Create a Watch Folder Input Filter • POST: Create a WebDAV Input Filter • POST: Create a VOD Catalog Input Filter

The RTMP `filter_settings` Element

The following table describes the settings for an RTMP input filter.

Tag	Type	Description
`automatically_promote_after_ingest`	boolean	For use only with VOD content.

Tag	Type	Description
		When **`true`**, Delta automatic ally converts VOD content to VOD catalog content upon completion.
`aws_credential_id`	integer	This field applies only if `storage_location` is an S3 URL. Enter the Amazon Web Service credential to use in order to be authenticated on the S3 server.

Tag	Type	Description
`content_window_type`	string	Specifies how long to store content: • `keep_seconds` : Enter the duration in `seconds_to_keep` . Delta saves content up to this limit. As new content is added, Delta removes old content on a first in, first out basis. If you plan to create a passthrough output filter for the content associated with this input filter, then make sure that this `content_window_type` is equal to or greater than the window of the upstream encoder/packager. Otherwise, playback can be disrupted. • `keep_all`: Never automatically discard content. This option is intended for short-lived content such as events or VOD content. When you're no longer using the content object and you delete it, Delta also deletes the stored content. This option is not recommended for

Tag	Type	Description
		linear sources because it eventually uses up all of your storage. • `packager_controlled` : WebDAV only. Content can be deleted by sending a DELETE request to the WebDAV server. The DELETE is sent from outside Delta, from your upstream packager that is posting to the WeDAV server on Delta. If you plan to create a live-to-VOD output filter for the content associated with this input filter, then the upstream encoder must be configured to retain content that is at least the same size as that intended VOD clip. For all values, keep in mind that Delta continues to ingest and store content, even if you later remove all output filters from this content object.

Tag	Type	Description
`input_user_id`	integer	The ID of the WebDAV or RTMP user (created by POST Input User). When you post the content to the Delta WebDAV server, you must pass this ID and its corresponding password.
`listening_port`	integer	The port where Delta listens for incoming RTMP content. Default: 1935
`promotion_delay_seconds`	integer	When `automatically_promote_after_ingest` is **`true`**, `promotion_delay_seconds` indicates how many seconds the system waits after initial ingest to convert VOD content to VOD catalog content.
`seconds_to_keep`	integer	The number of seconds of content to keep on disk. Required if content_window_type is set to keep_seconds; ignored otherwise.

Tag	Type	Description
`storage_file_duration`	integer	Optional. When you use local storage, use this setting to customize the segment duration (in seconds) Delta uses when saving files to disk. The value must be greater than the source segment duration. If no value is entered, the duration on disk matches the source segment duration.

Tag	Type	Description

`storage_location`	string	The folder where Delta stores incoming content. Either:
		• A local directory that the Delta node has access to. Choose from one of the following folders or a folder under one of these. If the subfolder does not yet exist, Delta automatically creates it.
		  • `mnt`
		  • `tmp`
		  • `local_sources`
		  • `server`
		  • `pgsql`
		• A remote server: Any remote servers you mount via Settings. Note that you typically don't specify remote server storage for remote input.
		  If you have deployed a Delta cluster (for redundancy), any remote server you specify should be mounted on both nodes.
		• An Amazon Web Services S3 URL. Enter the URL as s3://<full path>

Tag	Type	Description
`template_id`	integer	Optional. Do one of the following: • Specify a template ID to associate with this input filter. Delta automatically associates the output filters belonging to the specified template with the content when the content object is created. • Omit this tag. You manually add output filters to the content after it has been created. See also Working with Output Templates in AWS Elemental Delta.
`time_source`	string	The time source used for Live to VOD clipping. Tags for the `time_source` element vary according to the input filter. For more information, see The time_source element.
`udp_input`	object	See The udp_input element.

Tag	Type	Description
`vod_content`	boolean	This field applies only to input filters that handle assets that could be either a stream or VOD. • If the asset is a stream, set the value to **`false`**. (A value of true will give unpredictable results.) • If the asset is VOD, set the value to **`true`**. (A value of false will give unpredictable results.) Default is `false`.
`vod_catalog_output_aws_credential_id`	integer	This field applies only if `vod_catalog_output_location` is an S3 URL. Enter the Amazon Web Service credential to use in order to be authenticated on the S3 server.

Tag	Type	Description

`vod_catalog_output_location`	string	Folder to store VOD Catalog contents produced using a Live-to-VOD Catalog output filter. Either:
		• A local directory that the Delta node has access to - one of the following folders or a folder under one of these (if the subfolder does not yet exist, it will be automatically created):
		  • `mnt`
		  • `tmp`
		  • `local_sources`
		  • `server`
		  • `pgsql-delta-vod`
		  • `pgsql`
		• An Amazon Web Services S3 URL. Enter the URL as `s3://<full path>`

`vod_output_template_alias`	string	The alias of a VOD Catalog output template. Required on a VOD Catalog input filter and when you're converting VOD content to VOD catalog after ingest. For information about creating a VOD Catalog output template, see Working with Output Templates in AWS Elemental Delta.

Tag	Type	Description
`vod_url_prefix`	string	Optional. Partial URL path added to the start of all VOD Catalog content endpoints produced through the input filter.

The `time_source` Element

The following table describes the settings for the `time_source` element.

Tag	Type	Description
`ingest_time`	string	Time from the Delta node.
`embedded`	string	Timecode for the source video. Times are assumed to be Coordinated Universal Time (UTC). If `embedded` is used and is not available in the stream, `ingest_time` is used instead.

POST: Create a UDP Input Filter

To create a UDP input filter and ingest content, send a POST request. The following sections describe how to format the request and provide examples of the request and response.

HTTP URL

```
POST http://Delta IP address:8080/input_filters
```

Body of HTTP

The body of your request is XML content consisting of one `input_filter` element that contains the following:

- General settings.
- One `filter_settings` element.

  See The UDP `filter_settings` Element for tag details.

> **Note**
>
> A VOD Catalog output template is required when you create a VOD Catalog input filter, use VOD Catalog elements in other input filters, or if you promote any content to VOD Catalog after it has been ingested. For information about output templates, see Working with Output Templates in AWS Elemental Delta.
> For information about VOD catalog, see Working with VOD Catalog Assets in AWS Elemental Delta.

Request Example

This request adds a UDP input filter.

```
POST http:10.24.34.2:8080/input_filters
----------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<input_filters>
    <input_filter>
        <filter_type>udp_input</filter_type>
        <label>Channel News Live</label>
        <filter_settings>
            <content_window_type>keep_seconds</content_window_type>
            <seconds_to_keep>14400</seconds_to_keep>
            <storage_location>/data/mnt/storage/live/news/</storage_location>
            <segmentation_marker>idr</segmentation_marker>
            <enable_fec_rx>false</enable_fec_rx>
            <udp_input>
                <uri>udp://10.24.34.2:5001</uri>
                <igmp_source/>
            </udp_input>
            <udp_input>
```

```
                <uri>udp://10.24.34.2:5002</uri>
               <igmp_source>
            </udp_input>
           <udp_input>
               <uri>udp://10.24.34.2:5003</uri>
               <igmp_source/>
           </udp_input>
       </filter_settings>
   </input_filter>
</input_filters>
```

Response

The response repeats back the data that you posted, with the addition of unique IDs that Delta assigns to the `input_filter` and to the `filter_settings`.

The response is identical to the response to a GET Input Filter. SeeGET Input Filter for an example.

Topics

- The input_filter Element
- The UDP filter_settings Element

The `input_filter` Element

All input filters contain these tags. For convenience, they are repeated in each input filter section.

Tag	Type	Description
`filter_type`	string	One of these: • `http_put` • `mss_input` • `remote_input` • `rtmp_input` • `udp_input` • `watch_folder_input`

Tag	Type	Description
		• `webdav_input` • `vod_catalog_input`
`label`	string	Name of the filter. For a UDP input filter, Delta copies this value to the `name` tag of the content object associated with this input filter. If blank, Delta automatically generates the value as 'Filter *`[Delta-assigned ID number]`* '. For example, `Filter 8`.

Tag	Type	Description
`filter_settings`		A setting that holds tags applicable to the input filter types. For applicable tags by filter, see the following sections: • POST: Create an HTTP PUT Input Filter • POST: Create an MSS Input Filter • POST: Create a Remote Input Filter • POST: Create an RTMP Input Filter • POST: Create a UDP Input Filter • POST: Create a Watch Folder Input Filter • POST: Create a WebDAV Input Filter • POST: Create a VOD Catalog Input Filter

The UDP `filter_settings` Element

The following table describes the settings for a UDP input filter.

Tag	Type	Description
`automatically_promote_after_ingest`	boolean	For use only with VOD content.

Tag	Type	Description
		When **`true`**, Delta automatic ally converts VOD content to VOD catalog content upon completion.
`aws_credential_id`	integer	This field applies only if `storage_location` is an S3 URL. Enter the Amazon Web Service credential to use in order to be authenticated on the S3 server.

Tag	Type	Description
`content_window_type`	string	Specifies how long to store content: • `keep_seconds` : Enter the duration in `seconds_to_keep` . Delta saves content up to this limit. As new content is added, Delta removes old content on a first in, first out basis. If you plan to create a passthrough output filter for the content associated with this input filter, then make sure that this `content_window_type` is equal to or greater than the window of the upstream encoder/packager. Otherwise, playback can be disrupted. • `keep_all`: Never automatically discard content. This option is intended for short-lived content such as events or VOD content. When you're no longer using the content object and you delete it, Delta also deletes the stored content. This option is not recommended for

Tag	Type	Description
		linear sources because it eventually uses up all of your storage. • `packager_controlled` : WebDAV only. Content can be deleted by sending a DELETE request to the WebDAV server. The DELETE is sent from outside Delta, from your upstream packager that is posting to the WeDAV server on Delta. If you plan to create a live-to-VOD output filter for the content associated with this input filter, then the upstream encoder must be configured to retain content that is at least the same size as that intended VOD clip. For all values, keep in mind that Delta continues to ingest and store content, even if you later remove all output filters from this content object.

Tag	Type	Description
`enable_fec_rx`	boolean	• `true` means that SMPTE 2022-1 and SMPTE 2022-2 (ProMPEG) forward error correction (FEC) reception are enabled on the input stream. If FEC data is not received, the input functions, but Delta logs an error. Only compatible with RTP inputs. • `false` is the default. FEC reception is not enabled.
`promotion_delay_seconds`	integer	When `automatically_promote_after_ingest` is **`true`**, `promotion_delay_seconds` indicates how many seconds the system waits after initial ingest to convert VOD content to VOD catalog content.
`seconds_to_keep`	integer	The number of seconds of content to keep on disk. Required if content_window_type is set to keep_seconds; ignored otherwise.

Tag	Type	Description
`segmentation_marker`	string	A tag that identifies the type of segmentation marker in the input stream. • **`idr`**: Default. IDR segments at each closed GOP boundary. • **`ebr`**: segments at Encoder Boundary Points as specified by OpenCable OC-SP-EBP-I01-130118.
`storage_file_duration`	integer	Optional. When you use local storage, use this setting to customize the segment duration (in seconds) Delta uses when saving files to disk. The value must be greater than the source segment duration. If no value is entered, the duration on disk matches the source segment duration.

Tag	Type	Description

`storage_location`	string	The folder where Delta stores incoming content. Either:
		• A local directory that the Delta node has access to. Choose from one of the following folders or a folder under one of these. If the subfolder does not yet exist, Delta automatically creates it.
		  • `mnt`
		  • `tmp`
		  • `local_sources`
		  • `server`
		  • `pgsql`
		• A remote server: Any remote servers you mount via Settings. Note that you typically don't specify remote server storage for remote input.
		  If you have deployed a Delta cluster (for redundancy), any remote server you specify should be mounted on both nodes.
		• An Amazon Web Services S3 URL. Enter the URL as s3://<full path>

Tag	Type	Description
`template_id`	integer	Optional. Do one of the following: • Specify a template ID to associate with this input filter. Delta automatically associates the output filters belonging to the specified template with the content when the content object is created. • Omit this tag. You manually add output filters to the content after it has been created. See also Working with Output Templates in AWS Elemental Delta.
`time_source`	string	The time source used for Live to VOD clipping. Tags for the `time_source` element vary according to the input filter. For more information, see The time_source element.

Tag	Type	Description
`vod_catalog_output_aws_credential_id`	integer	This field applies only if `vod_catalog_output_location` is an S3 URL. Enter the Amazon Web Service credential to use in order to be authenticated on the S3 server.

`vod_catalog_output_location`	string	Folder to store VOD Catalog contents produced using a Live-to-VOD Catalog output filter. Either:
		• A local directory that the Delta node has access to - one of the following folders or a folder under one of these (if the subfolder does not yet exist, it will be automatically created):
		  • `mnt`
		  • `tmp`
		  • `local_sources`
		  • `server`
		  • `pgsql-delta-vod`
		  • `pgsql`
		• An Amazon Web Services S3 URL. Enter the URL as `s3://<full path>`

Tag	Type	Description
`vod_output_template_alias`	string	The alias of a VOD Catalog output template. Required on a VOD Catalog input filter and when you're converting VOD content to VOD catalog after ingest. For information about creating a VOD Catalog output template, see Working with Output Templates in AWS Elemental Delta.
`vod_url_prefix`	string	Optional. Partial URL path added to the start of all VOD Catalog content endpoints produced through the input filter.

The `time_source` Element

The following table describes the settings for the `time_source` element.

Tag	Type	Description
`ingest_time`	string	Time from the Delta node.
`embedded`	string	Timecode for the source video. Times are assumed to be Coordinated Universal Time (UTC).

Tag	Type	Description
		If `embedded` is used and is not available in the stream, `ingest_time` is used instead.

The `udp_input` Element

The following table describes the settings for the `udp_input` element.

Tag	Type	Description
`uri`	string	The URI of the UDP or RTP asset to ingest. Format: `<protocol>://<IP address>:<port>` Example: `udp://239.255.1.10:5001` If the firewall is enabled, make sure this port is on the list of open incoming ports (**Settings** > **Firewall**).
`igmp_source`	string	Optional. The IP address for source-specific multicast streams.
`interface`	string	Optional. Network interface to use (such as `eth2`). If not specified, Delta uses the system routing table to select an interface.

Tag	Type	Description
`program_number`	integer	For use with Multi-program Transport Streams (MPTS) to indicate which program's tracks Delta ingests.

POST: Create a Watch Folder Input Filter

To create a Watch Folder input filter and ingest content, send a POST request. The following sections describe how to format the request and provide examples of the request and response.

HTTP URL

```
POST http://Delta IP address:8080/input_filters
```

Body of HTTP

The body of your request is XML content consisting of one `input_filter` element that contains the following:

- General settings.
- One `filter_settings` element.

  See The Watch Folder `filter_settings` Element for tag details.

> **Note**
>
> A VOD Catalog output template is required when you create a VOD Catalog input filter, use VOD Catalog elements in other input filters, or if you promote any content to VOD Catalog after it has been ingested. For information about output templates, see Working with Output Templates in AWS Elemental Delta.
> For information about VOD catalog, see Working with VOD Catalog Assets in AWS Elemental Delta.

Request Example

See the UDP Request Example for a general input filter request.

Response

The response repeats back the data that you posted, with the addition of unique IDs that Delta assigns to the `input_filter` and to the `filter_settings`.

The response is identical to the response to a GET Input Filter. SeeGET Input Filter for an example.

Topics

- The input_filter Element
- The Watch Folder filter_settings Element

The `input_filter` Element

All input filters contain these tags. For convenience, they are repeated in each input filter section.

Tag	Type	Description
`filter_type`	string	One of these: • `http_put` • `mss_input` • `remote_input` • `rtmp_input` • `udp_input` • `watch_folder_input` • `webdav_input` • `vod_catalog_input`
`label`	string	Name of the filter. For a UDP input filter, Delta copies this value to the `name` tag of the content object associated with this input filter. If blank, Delta automatically generates

Tag	Type	Description
		the value as 'Filter *[Delta-assigned ID number]* '. For example, `Filter 8`.
`filter_settings`		A setting that holds tags applicable to the input filter types. For applicable tags by filter, see the following sections: • POST: Create an HTTP PUT Input Filter • POST: Create an MSS Input Filter • POST: Create a Remote Input Filter • POST: Create an RTMP Input Filter • POST: Create a UDP Input Filter • POST: Create a Watch Folder Input Filter • POST: Create a WebDAV Input Filter • POST: Create a VOD Catalog Input Filter

The Watch Folder `filter_settings` Element

The following table describes the settings for a Watch Folder input filter.

Tag	Type	Description
`depth`	integer	The number of layers of subdirectories to monitor inside the watch folder. • **`0`** (default): Monitor only the top-level folder. • Nil: Recursively monitor all subdirectories in the folder. • 1-10: Monitor this number of layers.
`incoming`	location	The folder to watch, which must be accessible to Delta (for example, it must be a remote server mounted onto Delta via **Settings** > Mount Points). For contents of the location type, see Location Type Elements.
`search_subfolders`	boolean	Enable Delta to search subfolders. Delta applies the same output filters to all content in all subfolders.
`template_id`	integer	Optional. Do one of the following: • Specify a template ID to associate with this input filter. Delta automatically associates the output filters belonging to the specified template with the content

Tag	Type	Description
		when the content object is created. • Omit this tag. You manually add output filters to the content after it has been created. See also Working with Output Templates in AWS Elemental Delta.

POST: Create a WebDAV Input Filter

To create a WebDAV input filter and ingest content, send a POST request. The following sections describe how to format the request and provide examples of the request and response.

HTTP URL

```
POST http://Delta IP address:8080/input_filters
```

Body of HTTP

The body of your request is XML content consisting of one `input_filter` element that contains the following:

- General settings.
- One `filter_settings` element.

  See The WebDAV Folder `filter_settings` Element for tag details.

> **Note**
>
> A VOD Catalog output template is required when you create a VOD Catalog input filter, use VOD Catalog elements in other input filters, or if you promote any content to VOD Catalog after it has been ingested. For information about output templates, see Working with Output Templates in AWS Elemental Delta.

For information about VOD catalog, see Working with VOD Catalog Assets in AWS Elemental Delta.

Request Example

See the UDP Request Example for a general input filter request.

Response

The response repeats back the data that you posted, with the addition of unique IDs that Delta assigns to the `input_filter` and to the `filter_settings`.

The response is identical to the response to a GET Input Filter. SeeGET Input Filter for an example.

Topics

- The input_filter Element
- The WebDAV Folder filter_settings Element

The `input_filter` Element

All input filters contain these tags. For convenience, they are repeated in each input filter section.

Tag	Type	Description
`filter_type`	string	One of these: • `http_put` • `mss_input` • `remote_input` • `rtmp_input` • `udp_input` • `watch_folder_input` • `webdav_input` • `vod_catalog_input`

Tag	Type	Description
`label`	string	Name of the filter. For a UDP input filter, Delta copies this value to the `name` tag of the content object associated with this input filter. If blank, Delta automatically generates the value as 'Filter *`[Delta-assigned ID number]`* '. For example, `Filter 8`.
`filter_settings`		A setting that holds tags applicable to the input filter types. For applicable tags by filter, see the following sections: • POST: Create an HTTP PUT Input Filter • POST: Create an MSS Input Filter • POST: Create a Remote Input Filter • POST: Create an RTMP Input Filter • POST: Create a UDP Input Filter • POST: Create a Watch Folder Input Filter • POST: Create a WebDAV Input Filter • POST: Create a VOD Catalog Input Filter

The WebDAV Folder `filter_settings` Element

The following table describes the settings for a WebDAV input filter.

Tag	Type	Description
`automatically_promote_after_ingest`	boolean	For use only with VOD content. When **`true`**, Delta automatically converts VOD content to VOD catalog content upon completion.
`aws_credential_id`	integer	This field applies only if `storage_location` is an S3 URL. Enter the Amazon Web Service credential to use in order to be authenticated on the S3 server.
`content_window_type`	string	Specifies how long to store content: • `keep_seconds` : Enter the duration in `seconds_to_keep` . Delta saves content up to this limit. As new content is added, Delta removes old content on a first in, first out basis. If you plan to create a passthrough output filter for the content associated with this input filter, then make sure that this `content_window_typ`

Tag	Type	Description
		e is equal to or greater than the window of the upstream encoder/packager. Otherwise, playback can be disrupted. • `keep_all`: Never automatically discard content. This option is intended for short-lived content such as events or VOD content. When you're no longer using the content object and you delete it, Delta also deletes the stored content. This option is not recommended for linear sources because it eventually uses up all of your storage. • `packager_controlled`: WebDAV only. Content can be deleted by sending a DELETE request to the WebDAV server. The DELETE is sent from outside Delta, from your upstream packager that is posting to the WeDAV server on Delta. If you plan to create a live-to-VOD output filter for

Tag	Type	Description
		the content associated with this input filter, then the upstream encoder must be configured to retain content that is at least the same size as that intended VOD clip. For all values, keep in mind that Delta continues to ingest and store content, even if you later remove all output filters from this content object.
`encrypt_storage`	boolean	When **`true`**, AWS Elemental Delta encrypts ingested content on storage.
`full_url`	string	The primary node's absolute path using the node's hostname.
`input_user_id`	integer	The ID of the WebDAV or RTMP user (created by POST Input User). When you post the content to the Delta WebDAV server, you must pass this ID and its corresponding password.

Tag	Type	Description
`promotion_delay_seconds`	integer	When `automatically_promote_after_ingest` is **`true`**, `promotion_delay_seconds` indicates how many seconds the system waits after initial ingest to convert VOD content to VOD catalog content.
`public_endpoint`	string	The primary node's absolute path using the node's IP address.

Tag	Type	Description
`relative_uri`	string	For HTTP or MSS: • If content has been posted to <Delta IP address>: 8080/in_mss/ or ../in_http/, then leave blank. • If it has been posted to a subfolder of `<Delta IP address>:8080/in_mss/` or `../in_http` ., then specify that subfolder path in format `<subfolder/sub-subfolder/>` (no leading slash). For WebDAV: • If content has been posted to `<WebDAV server>/in/` , leave blank. • If it has been posted to a subfolder of `<WebDAV server>/in/` , then specify that subfolder path in format `<subfolder/sub-subfolder/>` (no leading slash).
`seconds_to_keep`	integer	The number of seconds of content to keep on disk. Required if content_window_type is set to keep_seconds; ignored otherwise.

Tag	Type	Description

`storage_location`	string	The folder where Delta stores incoming content. Either:
		• A local directory that the Delta node has access to. Choose from one of the following folders or a folder under one of these. If the subfolder does not yet exist, Delta automatically creates it.
		  • `mnt`
		  • `tmp`
		  • `local_sources`
		  • `server`
		  • `pgsql`
		• A remote server: Any remote servers you mount via Settings. Note that you typically don't specify remote server storage for remote input.
		  If you have deployed a Delta cluster (for redundancy), any remote server you specify should be mounted on both nodes.
		• An Amazon Web Services S3 URL. Enter the URL as s3://<full path>

Tag	Type	Description
`template_id`	integer	Optional. Do one of the following: • Specify a template ID to associate with this input filter. Delta automatically associates the output filters belonging to the specified template with the content when the content object is created. • Omit this tag. You manually add output filters to the content after it has been created. See also Working with Output Templates in AWS Elemental Delta.
`time_source`	string	The time source used for Live to VOD clipping. Tags for the `time_source` element vary according to the input filter. For more information, see The time_source element.
`udp_input`	object	See The udp_input element.
`vips`	integer	The absolute paths for the virtual IP addresses (VIPs) of leader and secondary Delta nodes in a cluster.

Tag	Type	Description
`vod_content`	boolean	This field applies only to input filters that handle assets that could be either a stream or VOD. • If the asset is a stream, set the value to **`false`**. (A value of true will give unpredictable results.) • If the asset is VOD, set the value to **`true`**. (A value of false will give unpredictable results.) Default is `false`.
`vod_catalog_output_aws_credential_id`	integer	This field applies only if `vod_catalog_output_location` is an S3 URL. Enter the Amazon Web Service credential to use in order to be authenticated on the S3 server.

Tag	Type	Description

`vod_catalog_output_location`	string	Folder to store VOD Catalog contents produced using a Live-to-VOD Catalog output filter. Either:
		• A local directory that the Delta node has access to - one of the following folders or a folder under one of these (if the subfolder does not yet exist, it will be automatically created):
		  • `mnt`
		  • `tmp`
		  • `local_sources`
		  • `server`
		  • `pgsql-delta-vod`
		  • `pgsql`
		• An Amazon Web Services S3 URL. Enter the URL as `s3://<full path>`

`vod_output_template_alias`	string	The alias of a VOD Catalog output template. Required on a VOD Catalog input filter and when you're converting VOD content to VOD catalog after ingest. For information about creating a VOD Catalog output template, see Working with Output Templates in AWS Elemental Delta.

Tag	Type	Description
`vod_url_prefix`	string	Optional. Partial URL path added to the start of all VOD Catalog content endpoints produced through the input filter.

The `time_source` Element

The following table describes the settings for the `time_source` element.

Tag	Type	Description
`ingest_time`	string	Time from the Delta node.
`embedded`	string	Timecode for the source video. Times are assumed to be Coordinated Universal Time (UTC). If `embedded` is used and is not available in the stream, `ingest_time` is used instead.
`program_date_time`	string	Time from the external source. If `program_date_time` is used and is not available in the stream, `ingest_time` is used instead.

The `udp_input` Element

The following table describes the settings for the `udp_input` element.

Tag	Type	Description
`uri`	string	The URI of the UDP or RTP asset to ingest. Format: `<protocol>://<IP address>:<port>` Example: `udp://239.255.1.10:5001` If the firewall is enabled, make sure this port is on the list of open incoming ports (**Settings** > **Firewall**).
`igmp_source`	string	Optional. The IP address for source-specific multicast streams.
`interface`	string	Optional. Network interface to use (such as `eth2`). If not specified, Delta uses the system routing table to select an interface.
`program_number`	integer	For use with Multi-program Transport Streams (MPTS) to indicate which program's tracks Delta ingests.

POST: Create a VOD Catalog Input Filter

To create a VOD Catalog input filter and ingest content, send a POST request. The following sections describe how to format the request and provide examples of the request and response.

HTTP URL

```
POST http://Delta IP address:8080/input_filters
```

Body of HTTP

The body of your request is XML content consisting of one `input_filter` element that contains the following:

- General settings.
- One `filter_settings` element.

  See The VOD Catalog Folder `filter_settings` Element for tag details.

> **Note**
>
> A VOD Catalog output template is required when you create a VOD Catalog input filter, use VOD Catalog elements in other input filters, or if you promote any content to VOD Catalog after it has been ingested. For information about output templates, see Working with Output Templates in AWS Elemental Delta.
> For information about VOD catalog, see Working with VOD Catalog Assets in AWS Elemental Delta.

Request Example

See the UDP Request Example for a general input filter request.

Response

The response repeats back the data that you posted, with the addition of unique IDs that Delta assigns to the `input_filter` and to the `filter_settings`.

The response is identical to the response to a GET Input Filter. SeeGET Input Filter for an example.

Topics

- The input_filter Element
- The VOD Catalog Folder filter_settings Element

The `input_filter` Element

All input filters contain these tags. For convenience, they are repeated in each input filter section.

Tag	Type	Description
`filter_type`	string	One of these: • `http_put` • `mss_input` • `remote_input` • `rtmp_input` • `udp_input` • `watch_folder_input` • `webdav_input` • `vod_catalog_input`
`label`	string	Name of the filter. For a UDP input filter, Delta copies this value to the `name` tag of the content object associated with this input filter. If blank, Delta automatically generates the value as 'Filter *`[Delta-assigned ID number]`* '. For example, `Filter 8`.
`filter_settings`		A setting that holds tags applicable to the input filter types. For applicable tags by filter, see the following sections: • POST: Create an HTTP PUT Input Filter

Tag	Type	Description
		• POST: Create an MSS Input Filter • POST: Create a Remote Input Filter • POST: Create an RTMP Input Filter • POST: Create a UDP Input Filter • POST: Create a Watch Folder Input Filter • POST: Create a WebDAV Input Filter • POST: Create a VOD Catalog Input Filter

The VOD Catalog Folder `filter_settings` Element

The following table describes the settings for a VOD Catalog input filter.

Tag	Type	Description
`aws_credential_id`	integer	This field applies only if `storage_location` is an S3 URL. Enter the Amazon Web Service credential to use in order to be authenticated on the S3 server.
`encrypt_path`	string	When you provide a location, AWS Elemental Delta encrypts content at ingest and stores encrypted content in the specified location.

Tag	Type	Description
`location`	string	The source location for all assets that the filter ingests.
`take_ownership_of_content`	boolean	When **`true`**, Delta takes ownership of the content after ingest. When you delete VOD catalog content from Delta, the original data is also deleted.
`vod_output_template_alias`	string	The alias of a VOD Catalog output template. Required on a VOD Catalog input filter and when you're converting VOD content to VOD catalog after ingest. For information about creating a VOD Catalog output template, see Working with Output Templates in AWS Elemental Delta.
`vod_url_prefix`	string	Optional. Partial URL path added to the start of all VOD Catalog content endpoints produced through the input filter.

Outputting Content from AWS Elemental Delta

The following sections describe how to manage the resources that AWS Elemental Delta uses for packaging and delivering content. These resources include output filters and endpoints.

Topics

- Creating Output Filters
- Modifying Output Filters
- Viewing Filters List: GET Output Filters List
- Viewing a Live or VOD Filter: GET Output Filter
- Viewing VOD Catalog Endpoints: GET Endpoints
- Deleting an Output Filter: DELETE

Creating Output Filters

This discusses the ways that you can add output filters to a content entity.

Methods for Creating Output Filters

You can initially create output filters in several ways.

In all of the following methods, the new output filter must be one of the following:

- A new top-level filter for the associated content.
- A child of an existing output filter that is at the end of a branch.

> **Note**
>
> You can't insert an output filter between two existing output filters.

When you can use each method

Each method for creating output filters has rules about when it can be used. The following table indicates if the method can be used for *initial output filter creation* (adding output filters to a

content entity that doesn't have any output filters assigned) or for *adding output filters* (adding output filters to a content entity that has existing output filters).

Method	Initial Creation	Adding More Filters
One by one	Yes	Yes
Filter sets	Yes	Yes
`add_filters`	Yes, but using filter sets is better.	Yes
Output filter template	Yes	No

The following sections describe the methods in detail.

One by One

The one by one method allows you to create a single filter using PUT Output Filter commands. See the following sections for details on the commands for creating filters one by one:

- Creating Package Output Filters in AWS Elemental Delta
- Creating Access Restriction Filters in AWS Elemental Delta
- Creating Processing Output Filters in AWS Elemental Delta

Several Filters: Filter Sets

The Filter Sets method allows you to add all or part of a filter tree in one PUT command. For help creating filter sets, see Working with Filter Sets and Add Filters in AWS Elemental Delta.

Several Filters: Add Filters Command

The Add Filters Command method allows you to add one filter or add all or part of a filter tree using the PUT Add Filters command.

The key advantage to using this command compared to either adding filters one by one or using filter sets is that the response returns only the filters you added, not the entire filter tree. This short response can be useful if the filter tree is very big. For help using the PUT Add Filters command, see `PUT Add_Filters`.

Using Output Filter Template

The Output Filter Template method allows you to create multiple instances of the same filter. For help creating output templates, see Working with Output Templates in AWS Elemental Delta.

> **Note**
>
> Output templates require much planning and a complete understanding of how filters are created, so use them only when you are familiar with creating output filters.

Modifying Output Filters

This section lists information on how you can modify an existing filter. It assumes that you are familiar with the contents of the filter to modify and that you have read the information in the sections on output filters, as listed in the following sections:

- Creating Package Output Filters in AWS Elemental Delta
- Creating Access Restriction Filters in AWS Elemental Delta
- Creating Processing Output Filters in AWS Elemental Delta

Topics

- Modifying an Element
- Adding an Element
- Removing an Element
- Modify Tags in an Output Filter
- Modify Tags in a filter_settings
- Add a New stream_set
- Add a Track to a stream_set
- Modify Tags in a Track
- Modify Tags in a keyprovider_settings
- Add a nonce_key_pairs
- Add a nonce_key_pair

Modifying an Element

You can modify an output filter in one of these ways:

- Modify the individual tags in the filter.
- Modify a sub-element in a filter. For example, modify the `filter_settings` element inside the output filter.

You cannot modify a filter to change its position in the filter tree.Instead, delete and recreate the filter. If the changed filter has child filters, you have to recreate all the children (because deleting the parent filter also deletes the children). The best way to recreate the children is to use a filter set as described in Filter Sets.

Element	Rule to Modify
`filter`	Can modify any individual tag in the filter.
`filter_settings` in any type of output filter.	Can modify any individual tag.
`stream_set` in a `filter_settings` (in a package output filter).	Not applicable; there are no individual tags to modify.
`video_track` , `audio_track` , `subtitle_track` in a `stream_set` (in a package output filter)	Can modify one or more individual tags.
`keyprovider_settings` in a `filter_settings` (in a DRM output filter)	Can modify any individual tag.
`nonce_key_pairs` in a filter (in an authentication output filter)	**Cannot** change an existing `nonce_key_pairs` .
`nonce_key_pair` in a `nonce_key_pairs` in a filter (in an authentication output filter)	Not applicable; there is nothing to modify.

Adding an Element

You can add a sub-element in a filter. You can either add a sub-element when there was none before or you can add more instances of an existing sub-element. For example, you can add a stream set to a package output filter.

Element	Count Rule	Rule to Add
`filter`	-	The rules are dependent on the type of filter. See the following sections on creating output filters: • Creating Package Output Filters in AWS Elemental Delta • Creating Access Restriction Filters in AWS Elemental Delta • Creating Processing Output Filters in AWS Elemental Delta
`filter_settings` in any type of output filter	One	Not applicable; if there is a `filter_settings` , then it is required on creation and the maximum is 1.
`stream_set` in a `filter_settings` in a package output filter	One or more	Can add a `stream_set` .
`video_track` , `audio_track` , `subtitle_track` in a `stream_set` in a package output filter	Zero or more	Can add a track.

Element	Count Rule	Rule to Add
keyprovider_settings in a filter_settings in a DRM output filter	Zero or one	**Cannot** add a keyprovider_settings . Instead, you must recreate the entire filter_settings with the new, revised content. All sub-elements get new IDs.
nonce_key_pairs in an authentication output filter	Zero or more	Can add a nonce_key_pairs .
nonce_key_pair in a nonce_key_pairs in an authentication output filter	One or more	Can add a nonce_key_pair .

Removing an Element

You cannot remove a sub-element inside a filter. For example, you cannot delete an unwanted stream set. Instead, delete and recreate the filter.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content id/filters/filter id
```

where:

- *content id* is the unique ID or alias of the content that holds the output filter to modify.
- *filter id* is the unique ID of the output filter to modify.

> **Note**
>
> The PUT to modify an existing output filter includes the filter ID in the HTTP URL while the PUT to create a new output filter does not include the filter ID.

Body of HTTP

The body contains the data to modify, change, or delete, as follows.

You may want to consult the diagrams in AWS Elemental Delta XML Structure, which show the basic structure of various output filters.

> **Note**
>
> A key difference in the body between a PUT to create an output filter and a PUT to modify an output filter is:
>
> - For a create, the body includes the `content` tag.
> - For a modify, the body does not include the `content` tag. It starts with the `filter` tag. Likewise, the response omits the entire content: it returns only the `filter` element and its tags.

See the following sections for information on the body to include for the modification you want to make.

Modify Tags in an Output Filter

You can send a PUT request to modify most of the *individual tags* in an output filter. An individual tag is a tag directly under the output filter, not inside a sub-element of the output filter. You cannot change individual tags that control other sub-elements; for example, you cannot change the `filter_type` because doing so could invalidate the tags in the `filter_settings` element.

You also cannot:

- Change the ID (REST ID) of any element.
- Change the `parent_id`, which is equivalent to moving the filter in the filter tree.
- Change the `filter_type`. Instead, delete and recreate the filter. If the changed filter has child filters, you have to recreate all the children (because deleting the parent filter also deletes the children). The best way to recreate the children is to use a filter set, as described in Working with Filter Sets and Add Filters in AWS Elemental Delta.

The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/contents_ID/filters/filters_ID
```

Body of HTTP Request

The XML body consists of one `filter` element containing:

- The tag or tags to change.

Request Example

This request sets the endpoint tag to true in the output filter that has the ID 21.

```
PUT http://10.24.34.2:8080/contents/62/filters/21
  --------------------------------------------------
   <filter>
   <endpoint>true</endpoint>
   </filter>
```

Response

The response repeats back all the individual tags of the filter.

```
<?xml version="1.0" encoding="UTF-8"?>
  <filters href="/contents/39/filters" product="Delta" version="2.3.0.123456">
   <filter href="/contents/39/filters/21" product="Delta" version="2.3.0.123456">
   <id>21</id>
   <filter_type>live_to_vod</filter_type>
   <endpoint>true</endpoint>
   output_url903</output_url>
   <filter_type>mpeg_ts_package</filter_type>
   </filter>
  </filters>
```

Modify Tags in a `filter_settings`

You can send a PUT request to modify the value of any tag in a `filter_settings`. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/contents_ID/filters/filters_ID
```

Body of HTTP Request

The XML body consists of:

One `filters` container and one filter element containing:

- The ID of the filter.
- One `filter_settings` element containing:
  - The ID of the `filter_settings`.
  - Only the tag or tags to change.

Request Example

This request changes the `filter_settings` that belongs to the output filter that has the ID 21. It changes the `index_duration` to 90.

```
PUT http://10.24.34.2:8080/contents/39/filters/21
  --------------------------------------------------
  <filters>
   <filter>
   <id>21</id>
   <filter_settings>
   <id>15</id>
   <index_duration>90</index_duration>
   </filter_settings>
   </filter>
  </filters>
```

Response

The response repeats back all the individual tags of the `filter_settings`. It also includes the individual tags in the filter but omits other sub-elements of the filter, such as the `stream_set` elements.

```
<?xml version="1.0" encoding="UTF-8"?>
  <filters href="/contents/39/filters" product="Delta" version="2.3.0.123456">
```

```
  <filters href="/contents/39/filters/21" product="Delta" version="2.3.0.123456">
  <id>15</id>
  <parent_id/>
  <endpoint>false</endpoint>
  <output_url/>
  <filter_type>hds_package</filter_type>
  <use_default_stream_sets>true</use_default_stream_sets>
  <filter_settings>
  <id>15</id>
  <fragment_duration>2</fragment_duration>
  <index_duration>90</index_duration>
  <playlist_type/>
  <avail_trigger>all</avail_trigger>
  <ad_markers>none</ad_markers>
  <broadcast_time>false</broadcast_time>
  <ignore_web_delivery_allowed>false</ignore_web_delivery_allowed>
  <ignore_no_regional_blackout>false</ignore_no_regional_blackout>
  <enable_blackout>false</enable_blackout>
  <enable_network_end_blackout>false</enable_network_end_blackout>
  <network_id/>
  </filter_settings>
  </filter>
 </filters>
```

Add a New stream_set

You can send a PUT request to add a `stream_set` to a package output filter, assuming that adding this follows the rules for this type of package output filter (for example, the filter allows more than one `stream_set`). The following sections describe how to format the request.

> **Note**
>
> You can only add a new `stream_set` if `use_default_stream_sets` is false for this filter. If this tag is true, then `stream_sets` cannot be changed in any way: changes via the REST API are ignored.

HTTP URL

```
PUT http://Delta IP address:8080/contents/contents_ID/filters/filters_ID
```

Body of HTTP Request

The XML body consists of one `filter` element containing:

- The ID of the filter.
- One `stream_sets` container and one `stream_set` element containing:
  - Tags for the `stream_set`.
  - One each of `video_tracks`, `audio_tracks`, and `subtitle_tracks` containers holding:
    - `video_track` and/or `audio_track` and/or `subtitle_track` elements, in accordance with the rules for this type of output filter.

Request Example

This request adds a `stream_set` to the output filter that has the ID 21. This `output_filter` is an MPEG-TS filter, so it can take any number of `stream_sets`, each with one `video_track` and one `audio_track` (but it cannot take a `subtitle_track`).

Notice that you omit ID tags from the `stream_set`, `video_track,` and `audio_track` elements. In this way, the system adds the `stream_set` as a new element.

```
PUT http://10.24.34.2:8080/contents/39/filters/21
  --------------------------------------------------
   <filter>
   <id>21</id>
   <stream_sets>
   <stream_set>
   <stream_index>9</stream_index>
   <video_tracks>
   <video_track>
   <track_index>6</track_index>
   <track_id>7</track_id>
   </video_track>
   </video_tracks>
   <audio_tracks>
   <audio_track>
   <track_index>8</track_index>
   <track_id>6</track_id>
   </audio_track>
   </audio_tracks>
   </stream_set>
   </stream_sets>
   </filter>
```

Response

The response returns all the contents of the output filter. The new `stream_set` and its `video_track` and `audio_track` elements are all assigned unique IDs.

```
<?xml version="1.0" encoding="UTF-8"?>
  <filters href="/contents/39/filters" product="Delta" version="2.3.0.123456">
   <filter href="/contents/39/filters/31" product="Delta" version="2.3.0.123456"
   <id>21</id>
   <parent_id>4</parent_id>
   <endpoint>true</endpoint>
   <default_endpoint_uri>http://10.24.34.2/out/i/31.ts</default_endpoint_uri>
   <custom_endpoint_uri>http://10.24.34.2/out/u/903.ts</custom_endpoint_uri>
   output_url903</output_url>
   <filter_type>mpeg_ts_package</filter_type>
   <filter_settings>
   <id>1</id>
   .
   .
   .
   </filter_settings>
   <stream_sets>
   <stream_set>
   <id>33</id>
   <stream_index>9</stream_index>
   <iframe_only>false</iframe_only>
   <video_tracks>
   <video_track>
   <id>60</id>
   <track_index>6</track_index>
   <track_id>7</track_id>
   </video_track>
   </video_tracks>
   <audio_tracks>
   <audio_track>
   <id>61</id>
   <track_index>8</track_index>
   <track_id>6</track_id>
   <audio_track>
   </audio_tracks>
   </stream_set>
   <stream_set>
   <id>31</id>
   .
```

```
    .
    .
    </stream_set>
    </stream_sets>
    </filter>
  </filters>
```

Add a Track to a `stream_set`

You send a PUT request to add one or more `video_track`, `audio_track`, or `subtitle_track` elements to an existing `stream_set`, assuming that the addition follows the rules for this type of package output filter. The following sections describe how to format the request.

> **Note**
>
> You can only add tracks to a `stream_set` if `use_default_stream_sets` is false for this filter. If this tag is true, then you cannot change `stream_sets` in any way.

HTTP URL

```
PUT http://Delta IP address:8080/contents/contents_ID/filters/filters_ID
```

Body of HTTP Request

The XML body consists of one `filter` element containing:

- The ID of the filter.
- One `stream_sets` container and one `stream_set` element containing:
  - The ID of the `stream_set`.
  - One each of `video_tracks`, `audio_tracks`, and `subtitle_tracks` containers holding:
    - `video_track` and/or `audio_track` and/or `subtitle_track` elements, in accordance with the rules for this type of output filter.

Request Example

This request adds two more `audio_track` elements to the `stream_set` that was added in the previous example.

Notice that you must include the ID of the `stream_set`, but not IDs, for the new tracks. In this way, the system adds the tracks.

```
PUT http://10.24.34.2:8080/contents/39/filters/21
  --------------------------------------------------
   <filter>
   <id>21</id>
   <stream_set>
   <id>33</id>
   <audio_track>
   <track_index>9</track_index>
   <track_id>4</track_id>
   </audio_track>
   <audio_track>
   <track_index>10</track_index>
   <track_id>8</track_id>
   </audio_track>
   </stream_set>
   </filter>
```

Response

The response returns all the contents of the output filter. The new `audio_track` elements are assigned unique IDs.

```
<?xml version="1.0" encoding="UTF-8"?>
  <filters href="/contents/39/filters" product="Delta" version="2.3.0.123456">
   <filter href="/contents/39/filters/31" product="Delta" version="2.3.0.123456">
   <id>21</id>
   .
   .
   .
   <filter_settings>
   <id>1</id>
   .
   .
   .

   </filter_settings>
   <stream_sets>
   <stream_set>
   <id>33</id>
   <stream_index>9</stream_index>
```

```
  <iframe_only>false</iframe_only>
  <video_tracks>
  <video_track>
  <id>60</id>
  <track_index>6</track_index>
  <track_id>7</track_id>
  </video_track>
  </video_tracks>
  <audio_tracks>
  <audio_track>
  <id>61</id>
  <track_index>8</track_index>
  <track_id>6</track_id>
  </audio_track>
  <audio_track>
  <id>62</id>
  <track_index>9</track_index>
  <track_id>4</track_id>
  </audio_track>
  <audio_track>
  <id>63</id>
  <track_index>10</track_index>
  <track_id>8</track_id>
  </audio_track>
  </audio_tracks>
  </stream_set>
  <stream_set>
  <id>31</id>
  .
  .
  .
  </stream_set>
  </stream_sets>
  </filter>
 </filters>
```

Modify Tags in a Track

You can send a PUT request to modify the `track_index` tag in a `video_track`, `audio_track`, or `subtitle_track`.

You can only change the `track_index` if `use_default_stream_sets` is false for this filter. If this tag is true, then tracks cannot be changed in any way.

The following sections describe how to format the PUT request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/contents_ID/filters/filters_ID
```

Body of HTTP Request

The XML body consists of one `filter` element containing:

- The ID of the filter
- One `stream_sets` container and one `stream_set` element containing:
  - The ID of the `stream_set`.
  - One of `video_tracks`, `audio_tracks`, or `subtitle_tracks` containers holding:
    - `video_track` and/or `audio_track` and/or `subtitle_track` elements, in accordance with the rules for this type of output filter. Each element contains:
      - An `id` tag.
      - The tag to change.

Request Example

This request modifies the `track_index` of the `audio_track` that has the ID 62. It changes its index to 1, moving it to a higher priority.

Notice that you must include the ID of the `stream_set` and the track to modify. In this way, the system modifies the existing tracks rather than adding a new track.

```
PUT http://10.24.34.2:8080/contents/39/filters/21
--------------------------------------------------
<filter>
  <stream_sets>
    <stream_set>
      <id>33</id>
      <audio_tracks>
        <audio_track>
          <id>62</id>
          <track_index>1</track_index>
```

```
        </audio_track>
      </audio_tracks>
    </stream_set>
  </stream_sets>
</filter>
```

Response

The response returns all the contents of the output filter.

```
<?xml version="1.0" encoding="UTF-8"?>
<filters href="/contents/39/filters" product="Delta" version="2.3.0.123456">
  <filter href="/contents/39/filters/31" product="Delta" version="2.3.0.123456">
    <id>21</id>
    .
    .
    .
    <filter_settings>
      <id>1</id>
      .
      .
      .
    </filter_settings>
    <stream_sets>
      <stream_set>
        <id>33</id>
        <stream_index>9</stream_index>
        <iframe_only>false</iframe_only>
        <video_tracks>
          <video_track>
            <id>60</id>
            <track_index>6</track_index>
            <track_id>7</track_id>
          </video_track>
        </video_tracks>
        <audio_tracks>
          <audio_track>
            <id>61</id>
            <track_index>8</track_index>
            <track_id>6</track_id>
          </audio_track>
          <audio_track>
            <id>62</id>
            <track_index>1</track_index>
```

```
            <track_id>4</track_id>
          </audio_track>
          <audio_track>
            <id>63</id>
            <track_index>10</track_index>
            <track_id>8</track_id>
          </audio_track>
        </audio_tracks>
      </stream_set>
      <stream_set>
        <id>31</id>
        .
        .
        .
      </stream_set>
    </stream_sets>
  </filter>
</filters>
```

Modify Tags in a `keyprovider_settings`

You can send a PUT request to modify the tags in a `keyprovider_settings` element inside the `filter_settings` of a DRM output filter. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/contents_ID/filters/filters_ID
```

Body of HTTP Request

The XML body consists of one `filter` element containing:

- The ID of the filter
- One `filter_settings` element containing:
  - The ID of the `filter_settings`.
  - One `keyprovider_settings` containing:
    - An `id` tag.
    - The tags to change.

Request Example

This request changes the `request_cenc_key` tag inside an Irdeto `keyprovider_settings` element in the DRM output filter with the ID 14.

```
PUT http://10.24.34.2:8080/contents/39/filters/14
  ---------------------------------------------------
   <filter>
   <id>14</id>
   <filter_settings>
   <id>23</id>
      <keyprovider_settings>
        <id>1</id>
        <request_cenc_key>false</request_cenc_key>
      </keyprovider_settings>
   <filter_settings>
   </filter>
```

Add a `nonce_key_pairs`

You can send a PUT request add a `nonce_key_pairs` element (where previously there was none) to an authentication output filter. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/contents_ID/filters/filters_ID
```

Body of HTTP Request

The XML body consists of one `filter` element containing:

- The ID of the filter.
- One `filter_settings` element containing:
  - The ID of the `filter_settings`.
  - One `nonce_key_pairs` containing:
    - Once or more `nonce_key_pair` tags.

Request Example

```
PUT http://10.24.34.2:8080/contents/60/filters/40
```

```
  ---------------------------------------------------
   <filter>
   <id>40</id>
   <filter_settings>
   <nonce_key_pairs>
   <nonce_key_pair>
   <nonce>123456</nonce>
   <key>5c81fbf7cbcbc6efff2d79e4</key>
   </nonce_key_pair>
   </nonce_key_pairs>
   </filter_settings>
   </filter>
```

Response

The response returns all the contents of the output filter.

```
PUT http://10.24.34.2:8080/contents/60/filters/40
  ---------------------------------------------------
  <?xml version="1.0" encoding="UTF-8"?>
  <filters href="/contents/39/filters" product="Delta" version="2.3.0.123456">
   <filter href="/contents/39/filters/40" product="Delta" version="2.3.0.123456">
   <id>40</id>
   .
   .
   <filter_settings>
   <id>3</id>
   .
   .
   <nonce_key_pairs>
   <nonce_key_pair>
   <nonce>123456</nonce>
   <key>5c81fbf7cbcbc6efff2d79e4</key>
   </nonce_key_pair>
   </nonce_key_pairs>
   </filter_settings>
   </filter>
  </filters>
```

Add a nonce_key_pair

You can send a PUT request to add a nonce_key_pair element to an existing nonce_key_pairs element in an Authentication output filter. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/contents_ID/filters/filters_ID
```

Body of HTTP Request

The XML body consists of one filters container and one filter element containing:

- The ID of the filter.
- One filter_settings element containing:
  - The ID of the filter_settings.
  - One nonce_key_pairs containing:
    - One or more nonce_key_pair tags.

Request Example

```
PUT http://10.24.34.2:8080/contents/60/filters/40
  ---------------------------------------------------
  <filters>
  <filter>
   <id>40</id>
   <filter_settings>
   <nonce_key_pairs>
   <nonce_key_pair>
   <nonce>222222</nonce>
   <key>keyt7303d7668</key>
   </nonce_key_pair>
   </nonce_key_pairs>
   </filter_settings>
   </filters>
  </filter>
```

Response

The response returns all the contents of the output filter. Notice how the nonce_key_pairs has two nonce_key_pair elements: the one you added and the one that was already there.

```
PUT http://10.24.34.2:8080/contents/60/filters/40
  ---------------------------------------------------
  <?xml version="1.0" encoding="UTF-8"?>
  <filters href="/contents/39/filters" product="Delta" version="2.3.0.123456">
   <filter href="/contents/39/filters/40" product="Delta" version="2.3.0.123456">
   <id>40</id>
   .
   .
   .
   <filter_settings>
   <id>3</id>
   .
   .
   .
   <nonce_key_pairs>
   <nonce_key_pair>
   <nonce>123456</nonce>
   <key>5c81fbf7cbcbc6efff2d79e4</key>
   </nonce_key_pair>
   <nonce_key_pair>
   <nonce>222222</nonce>
   <key>keyt7303d7668</key>
   </nonce_key_pair>
   </nonce_key_pairs>
   </filter_settings>
   </filter>
  </filters>
```

Viewing Filters List: GET Output Filters List

To view the list of output filters associated with a specific content entity, send a GET request. The following sections describe how to format the request.

HTTP URL

```
GET http://Delta IP address:8080/contents/contents_ID/filters
```

where *contents_ID* is the unique ID of the content, obtained from GET Contents List.

HTTP URL for VOD Catalog Contents

Requests for information about VOD Catalog contents are formed differently, based on if you refer to the contents by content path (default) or by content alias (optional).

- Request URL using the content path

```
GET http://Delta IP address:8080/vod_catalog_contents/contents_path/filters
```

  where *contents_path* is the Delta-assigned ID of the content, obtained from GET VOD Catalog Contents List.

- Request URL using the content alias

```
GET http://Delta IP address:8080/vod_catalog_contents/valias/contents_alias/
filters
```

  where *contents_alias* is the alias that was defined in the Live to VOD Catalog output filter or in the VOD content promotion request. You can also obtain the alias from GET a VOD Catalog Content.

Response

XML content consisting of one `content` element that contains one `filters` container holding:

- One or more `filter` elements that each contains:
  - An HREF that specifies the path to the filter and the product and version installed on the node.
  - The tags and sub-elements of the filter. For examples of the filter contents, see the examples for the response to PUT to create an output filter in the output filter sections:
    - Creating Package Output Filters in AWS Elemental Delta
    - Creating Access Restriction Filters in AWS Elemental Delta
    - Creating Processing Output Filters in AWS Elemental Delta

```
GET http://10.24.34.2:8080/contents/9/filters
  ---------------------------------------------
  <?xml version="1.0" encoding="UTF-8"?>
  <filters>
   <filter href="/contents/9/filters/21" product="Delta" version="2.3.0.123456">
   <id>21</id>
   .
```

```
  .
  .
  </filter>
  <filter href="/contents/9/filters/22" product="Delta" version="2.3.0.123456">
  <id>22</id>
  .
  .
  .
  </filter>
  <filter href="/contents/9/filters/40" product="Delta" version="2.3.0.123456">
  <id>40</id>
  .
  .
  .
  </filter>
 </filters>
```

Viewing a Live or VOD Filter: GET Output Filter

To view one output filter that is associated with a specific live or video on demand (VOD) content entity, send a GET request. The following sections describe how to format the request.

HTTP Request and Response

HTTP URL

```
GET http://Delta IP address:8080/contents/contents_ID/filters/filter_ID
```

where *contents_ID* is the unique ID of the content, obtained from GET Contents List or GET a Content and Filters.

Response

XML content consisting of one `filter` element that contains:

- An HREF that specifies the path to the filter and the product and version installed on the node.
- The tags and sub-elements of the filter. For examples of the filter contents, see the examples for the response to PUT to create an output filter in the output filter sections:
  - Creating Package Output Filters in AWS Elemental Delta

- Creating Access Restriction Filters in AWS Elemental Delta
- Creating Processing Output Filters in AWS Elemental Delta

```
GET http://10.24.34.2:8080/contents/9/filters/21
  ---------------------------------------------------
  <?xml version="1.0" encoding="UTF-8"?>
  <filter href="/contents/9/filters/21" product="Delta" version="2.3.0.123456">
   <id>21</id>
   .
   .
   .
  </filter>
```

Viewing VOD Catalog Endpoints: GET Endpoints

To view the output filters that are marked as endpoints on a specific VOD Catalog content entity, send a GET request. The following sections describe how to format the request.

HTTP URL

```
GET http://Delta IP address:8080/vod_catalog_contents/ID/endpoints
```

where *ID* is the unique path of the content.

HTTP URL with Content Alias

```
GET http://Delta IP address:8080/vod_catalog_contents/valias/contents_alias/
endpoints
```

where *contents_alias* is the alias that was defined in the Live to VOD Catalog output filter or in the VOD content promotion request.

Response

XML content consisting of `filters` element that contain zero or more `filter` elements, each with:

- An HREF that specifies the unique ID of the content and the product and version installed on the node.

- The tags and sub-elements of the filter. For examples of the filter contents, see the examples for the response to PUT to create an output filter in the output filter sections:
  - Creating Package Output Filters in AWS Elemental Delta
  - Creating Access Restriction Filters in AWS Elemental Delta
  - Creating Processing Output Filters in AWS Elemental Delta

```
GET http://10.24.34.2:8080/vod_catalog_contents/valias/hlsvod2vod_hlsvod10/endpoints
  ----------------------------------------------
  <?xml version="1.0" encoding="UTF-8"?>
  <filters href="/vod_catalog_contents/valias/hlsvod2vod_hlsvod10/endpoints"
 product="Delta" version="2.3.0.123456">
   <filter href="/filters/28" product="Delta" version="2.3.0.123456">
   <id>28</id>
   .
   .
   .
   </filter>
   <filter href="/filters/10" product="Delta" version="2.3.0.123456">
   <id>10</id>
   .
   .
   .
   </filter>
  </filters>
```

Deleting an Output Filter: DELETE

To remove the specified output filter from the specified content entity, send a DELETE request. The following sections describe how to format the request.

The specified output filter and all its child output filters are deleted.

No other components of the content entity are deleted. In a package output filter, the `stream_sets` in the filter are deleted, but the tracks in the content entity are not deleted.

HTTP URL

```
DELETE http://Delta IP address:8080/contents/contents_ID/filters/filters_ID
```

Response

A 200 OK response indicates the delete was successful.

Creating Package Output Filters in AWS Elemental Delta

The following sections describe how to create package output filters. The package determines what types of playback devices can play the output content.

Topics

- PUT: Create a CMAF Package Output Filter
- PUT: Create a DASH-ISO Package Output Filter
- PUT: Create an HDS Package Output Filter
- PUT: Create an HLS Package Output Filter
- PUT: Create an MP4 Package Output Filter
- PUT: Create an MPEG-TS Package Output Filter
- PUT: Create an MSS Package Output Filter
- PUT: Create a Passthrough Package Output Filter
- Working with Tracks: GET Tracks

PUT: Create a CMAF Package Output Filter

To create a Common Media Application Format (CMAF) package output filter on a content object, send a PUT request. This filter outputs HLS fMP4 content. The following sections describe how to format the request and provide examples of the request and response.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID
```

Body of HTTP

The body of your request is XML content consisting of one `content` element that holds:

- One `filters` container and one `filter` element that holds several tags, and also holds:
  - One `filter_settings` element that holds several tags.
  - One `stream_sets` container and one or more `stream_set` elements, each with *one* of the following:

- One `video_tracks` container with one `video_track` element that holds several tags. No audio or subtitles.
- One `audio_tracks` container with one or more `audio_track` elements that hold several tags. This is an audio rendition group with no video or subtitles.
- One `subtitle_tracks` container with one or more `subtitle_track` elements that hold several tags. This is a subtitles rendition group with no audio or video.

Request Example

This request modifies the existing content entity that has ID 39 to add a CMAF output filter. The request omits the `parent_id`, which means it is a top-level filter. It uses the default stream sets.

```
PUT http://10.24.34.2:8080/contents/39
---------------------------------------
<content>
  <filters>
    <filter>
      <endpoint>true</endpoint>
      <filter_type>cmaf_package</filter_type>
      <use_default_stream_sets>true</use_default_stream_sets>
      <filter_settings>
        <segment_duration>2</segment_duration>
        <index_duration>60</index_duration>
        <playlist_type/>
        <avail_trigger>all</avail_trigger>
        <ad_markers>none</ad_markers>
        <broadcast_time>false</broadcast_time>
        <ignore_web_delivery_allowed>false</ignore_web_delivery_allowed>
        <ignore_no_regional_blackout>false</ignore_no_regional_blackout>
        <enable_blackout>false</enable_blackout>
        <enable_network_end_blackout>false</enable_network_end_blackout>
        <network_id/>
      </filter_settings>
    </filter>
  </filters>
</content>
```

Response

The response repeats back the filter that you created, with the addition of the following tags:

- `id`: Unique IDs for the filter and each sub-element as described in this section.

- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI, as follows:

  ```
  http://<Delta IP address>:8080/out/i/<filter ID>.m3u8
  ```

  where `filter ID` is the ID for this filter, assigned once the filter is created.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if `output_uri` contains a value. Contains the custom endpoint URI, as follows:

  ```
  http://<Delta IP address:8080>/out/u/<output_url>.m3u8
  ```

This example response is a representation. The `default_endpoint` now shows a value. The `filter_settings` shows the default values. The `filter`, `filter_settings`, `stream_set`, `video_track`, and `audio_track` elements are all assigned unique IDs. Your results may vary.

```
<?xml version="1.0" encoding="UTF-8"?>
<contents href="/contents" product="Delta" version="2.3.0.123456">
  <content href="/contents/95" product="Delta" version="2.3.0.123456">
    <id>95</id>
    <type>HLS</type>
    <name>hlscaptions1</name>
    <alias/>
    <path/>
    <total_filters>5</total_filters>
    <total_endpoints>5</total_endpoints>
    <vod_content>false</vod_content>
    <remote_input>false</remote_input>
    <output_template_id nil="true"/>
    <storage_location nil="true"/>
    <uri>hlscaptions1.m3u8</uri>
    <encrypted>false</encrypted>
    <vod>false</vod>
    <filters>
      <filter>
        <id>176</id>
        <label>Filter 176</label>
        <parent_id nil="true"/>
        <filter_type>cmaf_package</filter_type>
        <endpoint>true</endpoint>
        <default_endpoint_uri>http://10.24.34.2/out/i/176.m3u8</
default_endpoint_uri>
```

```
        <custom_endpoint_uri></custom_endpoint_uri>
        <ancestry nil="true"/>
        <url_extension>m3u8</url_extension>
        <output_url nil="true"/>
        <description nil="true"/>
        <use_default_stream_sets>true</use_default_stream_sets>
        <aliased_default_endpoint_uri></aliased_default_endpoint_uri>
        <aliased_custom_endpoint_uri></aliased_custom_endpoint_uri>
        <filter_settings>
          <stream_sets>
            <stream_set>
              <id>848</id>
              <stream_index>1</stream_index>
              <rendition_group_name nil="true"/>
              <iframe_only>false</iframe_only>
              <announced_bitrate nil="true"/>
              <video_tracks>
                <video_track>
                  <id>1456</id>
                  <track_index>0</track_index>
                  <track_id>135</track_id>
                  <rendition_group_value></rendition_group_value>
                  <track_selector nil="true"/>
                 </video_track>
              </video_tracks>
              <audio_tracks/>
              <subtitle_tracks/>
            </stream_set>
            <stream_set>
              <id>849</id>
              <stream_index>2</stream_index>
              <rendition_group_name nil="true"/>
              <iframe_only>false</iframe_only>
              <announced_bitrate nil="true"/>
              <video_tracks>
                <video_track>
                  <id>1457</id>
                  <track_index>1</track_index>
                  <track_id>140</track_id>
                  <rendition_group_value></rendition_group_value>
                  <track_selector nil="true"/>
                </video_track>
              </video_tracks>
              <audio_tracks/>
```

```
            <subtitle_tracks/>
          </stream_set>
          <stream_set>
            <id>850</id>
            <stream_index>3</stream_index>
            <rendition_group_name nil="true"/>
            <iframe_only>false</iframe_only>
            <announced_bitrate nil="true"/>
            <video_tracks>
              <video_track>
                <id>1458</id>
                <track_index>2</track_index>
                <track_id>139</track_id>
                <rendition_group_value></rendition_group_value>
                <track_selector nil="true"/>
              </video_track>
            </video_tracks>
            <audio_tracks/>
            <subtitle_tracks/>
        </stream_set>
        <stream_set>
          <id>851</id>
          <stream_index>4</stream_index>
          <rendition_group_name>audio</rendition_group_name>
          <iframe_only>false</iframe_only>
          <announced_bitrate nil="true"/>
          <video_tracks/>
          <audio_tracks>
            <audio_track>
              <id>1459</id>
              <track_index>3</track_index>
              <track_id>136</track_id>
              <rendition_group_value>audio</rendition_group_value>
              <track_selector nil="true"/>
            </audio_track>
          </audio_tracks>
          <subtitle_tracks/>
        </stream_set>
      </stream_sets>
    </filter>
  </filters>
 </content>
</contents>
```

Topics

- The filter Element
- The filter_settings Element
- The stream_set Element

The `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint.

Tag	Type	Description
		The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only. The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. Delta automatically creates this customized address, including the `output_url` . Delta stores the address in the `custom_endpoint_uri` tag. **Warning** If you have HLS and CMAF output filters on the same content, then you must make sure that the `output_url` value is unique. Otherwise, both filters will have the same URL and the endpoints will collide.

Tag	Type	Description
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.

Tag	Type	Description
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
`segment_duration`	integer	The length of TS segments that Delta generates. Delta rounds actual segments to hit the next available

Tag	Type	Description
		segmentation point in the source content. The default is the value from the ingested content.
`index_duration`	integer	The length of the live manifest (in seconds). Delta ignores this parameter for VOD content.
`playlist_type`	string	This tag specifies how Delta handles the `EXT-X-PLAYLIST-TYPE` tag in the output manifest for VOD and live-to-VOD content. • Empty (no value) is the default. Delta doesn't insert `EXT-X-PLAYLIST_TYPE` tags in the output. • **`EVENT`** means that Delta inserts the `EXT-X-PLAYLIST-TYPE:EVENT` in the manifest. • **`VOD`** means that Delta inserts the `EXT-X-PLAYLIST-TYPE:VOD` tag in the manifest.

Tag	Type	Description
`avail_trigger`	string	This tag specifies which type of ad avail SCTE-35 messages result in Delta including `cueInfo` tags in the output manifest. Typically, you should select the trigger to match the ad avail type that the input contains. • **`all`** is the default. The input uses splice inserts to identify ad avails. The input might also contain messages for other events such as chapters or programs. These events also result in `cueInfo` tags in the output manifest. • **`placement_opportunity`** indicates that the input uses time signals with a segmentation type of `placement opportunity`. The input might also contain messages for other events such as chapters or programs. These events also result in `cueInfo` tags in the output manifest.

Tag	Type	Description
`ad_markers`	string	This tag specifies how ad markers appear in the output. • **`none`** is the default and means that ad markers are not included in the output. • **`scte35_enhanced`** means that Delta generates ad markers and blackout tags based on SCTE-35 messages in the UDP or HLS input. • **`passthrough`** means that the output manifest contains a copy of the SCTE-35 ad markers from the input HLS manifest.
`broadcast_time`	boolean	This tag specifies how ID3 timed metadata messages appear in the output manifest. • **`false`** is the default. Delta does not insert messages in the output manifest. • **`true`** means that Delta generates messages every 5 seconds and inserts them in the output manifest with the content ingest time.

Tag	Type	Description
`ignore_web_delivery_allowed`	boolean	This tag specifies how Delta handles `web_delivery_allowed` tags from the input manifest. Applicable when `ad_markers` is **`scte35_enhanced`** only. • **`false`** is the default. SCTE-35 segmentation descriptors with `web_delivery_allowed` set to **`0`** trigger blackouts or avails. • **`true`** means that blackouts and avails are *not* triggered when `web_delivery_allowed` is set to **`0`**.

Tag	Type	Description
`ignore_no_regional_blackout`	boolean	This tag specifies how Delta handles `no_regional_blackout` tags in the input manifest. Applies when `ad_markers` is **`scte35_enhanced`** only. • **`false`** is the default. SCTE-35 segmentation descriptors with `no_regional_blackout` set to **`0`** trigger blackouts or avails. • **`true`** means that blackouts and avails are *not* triggered when `no_regional_blackout` is set to **`0`**.

Tag	Type	Description
`enable_blackout`	boolean	This tag specifies how `blackout` tags appear in the output manifest. Applicable when `ad_markers` is **`scte35_enhanced`** only. • **`false`** is the default and means that Delta does not insert `blackout` tags into the output manifest. • **`true`** means that Delta adds `blackout` tags to the output manifest based on SCTE-35 `Program`, `Chapter`, and `Not Indicated` segmentation descriptors when `web_delivery_allowed` or `no_regional_blackout` is set to **`0`**.

Tag	Type	Description
`enable_network_end_blackout`	boolean	This tag specifies how `blackout` tags appear in the output manifest. Applicable when `ad_markers` is **`scte35_enhanced`** only. • **`false`** is the default and means that Delta does not insert `blackout` tags into the output manifest. • **`true`** means that Delta adds `blackout` tags to the output manifest based on SCTE-35 `Network Start/End` segmentation descriptors. When **`true`**, you must also provide a value for `network_id` .
`network_id`	string	This tag specifies the EIDR ID of the network in the format 10.nnnn/xxxx-xxxx-xxxx-xxxx-xxxx-c (not case sensitive). Only network end events with this ID trigger blackout. Required when `enable_network_end_blackout` is **`true`**.

Tag	Type	Description
`include_program_date_time`	boolean	This tag specifies how Delta handles `EXT-X-PROGRAM-DATE-TIME` in the output manifest. • **`false`** is the default and means that Delta does not insert `EXT-X-PROGRAM-DATE-TIME` tags in the output manifest. • **`true`** means that Delta does insert `EXT-X-PROGRAM-DATE-TIME` tags in the output manifest, based on the input manifest. If the source content is HLS and it contains `EXT-X-PROGRAM-DATE-TIME` tags, then Delta uses the source time for the value. Otherwise, Delta uses the time from when the input filter processed the source segment.
`program_date_time_interval`	integer	When `include_program_date_time` is **`true`**, this tag specifies the frequency (in seconds) at which Delta inserts `EXT-X-PROGRAM-DATE-TIME` tags in the output manifest. The default is **`600`**.

The `stream_set` Element

The `stream_set` element holds information about how Delta handles the audio, video, and subtitles in a stream. The following table describes the settings for the `stream_set` element.

Tag	Type	Description
`stream_index`	integer	An optional number entered in each stream set in the filter to sort the sets in relation to each other. A lower number puts the set higher in the list. Numbers can be skipped. For example, if you later remove a stream set from the filter, there is no need to renumber the other sets.
`audio_tracks`	object	This tag contains informati on about the audio tracks to include in the stream set. To specify more than one audio track in the stream set, include multiple `audio_tra ck` objects. See the table below for the tags to enter in the `audio_track` .
`video_tracks`	object	This tag contains informati on about the video tracks to include in the stream set. To specify more than one video track in the stream set, include multiple `video_tra ck` objects. See the table

Tag	Type	Description
		below for the tags to enter in the `video_track` .
`subtitle_tracks`	object	This tag contains information about the subtitle or captions tracks to include in the stream set. To specify more than one subtitle or captions track in the stream set, include multiple `subtitle_track` objects. See the table below for the tags to enter in the `subtitle_track` .
`rendition_group_name`	string	A name entered to set this stream set up as a rendition group: all audio tracks are together in a rendition group. The video track works with any audio track in the rendition group. Leave blank to use a regular stream set.

Tag	Type	Description
`iframe_only`	boolean	This setting applies to stream sets with a single video track. The value is ignored in all other situations. • **`true`** means that the stream set will include an additional `EXT-X-I-FRAMES` tag. This tag prompts Delta to include an I-frame-only stream in the stream set with the other tracks that you specify. • **`false`** is the default and means that an I-frame-only stream is not included in the stream set.
`announced_bitrate`	boolean	Bitrate for the stream set. This value is used in adaptive bitrate bitrate manifest creation. If left blank, Delta auto-detects the value.

Tag	Type	Description
`audio_group_name`	string	This tag specifies what audio rendition group the video track uses. Options are: • Omit the tag: the default audio rendition group (the first one in the playlist) is used. • Include the tag with a rendition group name: the indicated audio rendition group is used. • Include the tag without a value (`audio_group_name/`): no audio rendition group is used. This is commonly used if the video track already has audio muxed in.

The `video_track`, `audio_track`, and `subtitle_track` Elements

The following table describes the settings for the `video_track`, `audio_track`, and `subtitle_track` elements.

Tag	Type	Description
`track_id`	integer	Read-only. The Delta-assigned numeric value for the track.
`track_index`	integer	A number used to sort the tracks in relation to each other if the stream set

Tag	Type	Description
		contains more than one track of the same type (video, audio, or subtitle). Enter a number in each *xx* track. A lower number puts the track higher in the list. Numbers can be skipped. For example, if you later remove a track from the set, there is no need to renumber the other tracks.
`rendition_group_value`	string	For audio and subtitle tracks only. A name assigned to each track if the stream set is a rendition group (you have defined a value for `rendition_group_name`). For example, assign a language designator.
`teletext_page_number`	hexadecimal string	Used on the `subtitle_track` only. The teletext page number within the incoming stream from which captions are extracted. Must be a three-digit hexadecimal string between the range of 100 and 8FF.

Tag	Type	Description
`language`	string	The language of the incoming track. Used on the `subtitle_track` only.

PUT: Create a DASH-ISO Package Output Filter

To create a DASH-ISO package output filter on a content object, send a PUT request. This filter outputs MP4 content. The following sections describe how to format the request and provide examples of the request and response.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID
```

Body of HTTP

The body of your request is XML content consisting of one `content` element that holds:

- One `filters` container and one `filter` element that holds several tags, and also holds:
  - One `filter_settings` element that holds several tags.
  - One `stream_sets` container and zero or one `stream_set` element (zero only if `use_default_stream_sets` is **`true`**). The `stream_set` element is required for DVB-Text and DVB-Sub subtitles.

    If present, the stream_set element holds several tags and also holds:

    - One `video_tracks` container and zero or one `video_track` element that holds several tags.
    - One `audio_tracks` container and zero or one `audio_track` element that holds several tags.
    - One `subtitle_tracks` container and zero or one `subtitle_track` element that holds several tags.

Request Example

This request modifies the existing content entity that has ID 39 to add a DASH output filter. The request omits the `parent_id`, which means it is a top-level filter. It includes an `output_url` (to create a custom endpoint that contains `movie21`). The `stream_set` is included to specify exactly which tracks to include.

```
PUT http://10.24.34.2:8080/contents/39
---------------------------------------
<content>
   <filters>
      <filter>
         <endpoint>true</endpoint><output_url>movie21</output_url>
         <filter_type>dash_iso_package</filter_type>
         <filter_settings>
            <fragment_duration>2</fragment_duration>
            <index_duration>60</index_duration>
            <profile/>
            <min_update_period>30</min_update_period>
            <min_buffer_time>20</min_buffer_time>
         </filter_settings>
         <stream_sets>
            <stream_set>
               <stream_index>0</stream_index>
               <video_tracks>
                  <video_track>
                     <track_index>0</track_index>
                     <track_id>54</track_id>
                  </video_track>
               </video_tracks>
               <audio_tracks>
                  <audio_track>
                     <track_index>1</track_index>
                     <track_id>55</track_id>
                  </audio_track>
               </audio_tracks>
            </stream_set>
         </stream_sets>
      </filter>
   </filters>
</content>
```

Response

The response repeats back the filter that you created, with the addition of the following tags:

- `id`: Unique IDs for the filter and each sub-element as described in this section.
- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI, as follows:

  ```
  http://<Delta IP address>:8080/out/i/<filter ID>.mpd
  ```

  where `filter ID` is the ID for this filter, assigned once the filter is created.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if `output_uri` contains a value. Contains the custom endpoint URI, as follows:

  ```
  http://<Delta IP address:8080>/out/u/<output_url>.mpd
  ```

This example response is a representation. The `default_endpoint` now shows a value. The `filter_settings` shows the default values. The `filter`, `filter_settings`, `stream_set`, `video_track`, and `audio_track` elements are all assigned unique IDs. Your results may vary.

```
<?xml version="1.0" encoding="UTF-8"?>
<contents href="/contents" product="Delta" version="2.3.0.123456">
  <content href="/contents/39" product="Delta" version="2.3.0.123456">
    .
    .
   <filters>
    <filter>
      <id>39</id>
      <label>Filter 597</label>
      <parent_id nil="true"/>
      <filter_type>dash_iso_package</filter_type>
      <endpoint>false</endpoint>
      <ancestry nil="true"/>
      <url_extension>mpd</url_extension>
      <output_url nil="true"/>
      <description nil="true"/>
      <use_default_stream_sets>true</use_default_stream_sets>
      <filter_settings>
        <id>41</id>
        <fragment_duration>2</fragment_duration>
        <index_duration>60</index_duration>
```

```
        <profile></profile>
        <multiperiod>false</multiperiod>
        <min_update_period>15</min_update_period>
        <min_buffer_time>30</min_buffer_time>
        <suggested_presentation_delay>25</suggested_presentation_delay>
      </filter_settings>
      <stream_sets>
        <stream_set>
          <id>4</id>
          <stream_index>0</stream_index>
          <iframe_only>false</iframe_only>
          <video_tracks>
            <video_track>
              <id>7</id>
              <track_index>0</track_index>
              <track_id>54</track_id>
            </video_track>
          </video_tracks>
          <audio_tracks>
            <audio_track>
              <id>10</id>
              <track_index>3</track_index>
              <track_id>56</track_id>
            </audio_track>
          </audio_tracks>
          <subtitle_tracks/>
        </stream_set>
      </stream_sets>
    </filter>
  </filters>
 </content>
</contents>
```

Topics

- The filterElement
- The filter_settings Element
- The stream_set Element

The `filterElement`

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only.

Tag	Type	Description
		The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. Delta automatically creates this customized address, including the `output_url` . Delta stores the address in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.

Tag	Type	Description
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
`fragment_duration`	integer	The length of fragments (in seconds) that Delta generates. Delta rounds actual fragments to the

Tag	Type	Description
		nearest multiple of the source content fragment length. The default is the value from the ingested content.
`index_duration`	integer	The length of the live manifest (in seconds). Delta ignores this parameter for VOD content.
`profile`	string	The DASH profile that Delta uses for the output. • Empty (no value) is the default. The output doesn't use a DASH profile. • **`hbbtv_1.5`** means that the output is compliant with HbbTV. • **`hybridcast`** means that the output is compliant with IPTV Forum Japan's Hybridcast system.
`min_update_period`	integer	The minimum time (in seconds) that the player must wait before requesting updates to the manifest.
`min_buffer_time`	integer	The minimum length of time (in seconds) that the player must keep content available in the buffer.

Tag	Type	Description
`multiperiod`	boolean	When **`true`**, AWS Elemental Delta breaks the output DASH manifest into multiple periods based on ad markers from the source content.
`suggested_presentation_delay`	integer	The minimum length of time (in seconds) that the player be from the end of the manifest.

The `stream_set` Element

The `stream_set` element holds information about how Delta handles the audio, video, and subtitles in a stream. The following table describes the settings for the `stream_set` element.

Tag	Type	Description
`stream_index`	integer	An optional number entered in each stream set in the filter to sort the sets in relation to each other. A lower number puts the set higher in the list. Numbers can be skipped. For example, if you later remove a stream set from the filter, there is no need to renumber the other sets.
`audio_tracks`	object	This tag contains informati on about the audio tracks to include in the stream set. To specify more than one audio track in the stream set,

Tag	Type	Description
		include multiple `audio_track` objects. See the table below for the tags to enter in the `audio_track` .
`video_tracks`	object	This tag contains information about the video tracks to include in the stream set. To specify more than one video track in the stream set, include multiple `video_track` objects. See the table below for the tags to enter in the `video_track` .
`subtitle_tracks`	object	This tag contains information about the subtitle or captions tracks to include in the stream set. To specify more than one subtitle or captions track in the stream set, include multiple `subtitle_track` objects. See the table below for the tags to enter in the `subtitle_track` .

The `video_track`, `audio_track`, and `subtitle_track` Elements

The following table describes the settings for the `video_track`, `audio_track`, and `subtitle_track` elements.

Tag	Type	Description
`track_id`	integer	Read-only.

Tag	Type	Description
		The Delta-assigned numeric value for the track.
`track_index`	integer	A number used to sort the tracks in relation to each other if the stream set contains more than one track of the same type (video, audio, or subtitle). Enter a number in each *xx* track. A lower number puts the track higher in the list. Numbers can be skipped. For example, if you later remove a track from the set, there is no need to renumber the other tracks.

PUT: Create an HDS Package Output Filter

To create an HTTP Dynamic Streaming (HDS) package output filter on a content object, send a PUT request. The following sections describe how to format the request and provide examples of the request and response.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID
```

Body of HTTP

The body of your request is XML content consisting of one `content` element that holds:

- One `filters` container and one `filter` element that holds several tags, and also holds:
  - One `filter_settings` element that holds several tags.

- One `stream_sets` container and zero or more `stream_set` elements (zero only if `use_default_stream_sets` is **`true`**). If present, each `stream_set` element holds several tags and also holds:
  - One `video_tracks` container and one `video_track` element that holds several tags.
  - One `audio_tracks` container and zero or more `audio_track` elements that hold several tags.

Request Example

This request modifies the existing content entity that has ID 39 to add an HDS output filter. The request omits the `parent_id`, which means it is a top-level filter. It includes an `output_url` (to create a custom endpoint that contains `movie21`).The `stream_set` is not specified because the default behavior is acceptable.

```
PUT http://10.24.34.2:8080/contents/39
---------------------------------------
<content>
  <filters>
    <filter>
      <endpoint>true</endpoint><output_url>movie21</output_url>
      <filter_type>hds_package</filter_type>
      <filter_settings>
        <fragment_duration>2</fragment_duration>
        <index_duration>60</index_duration>
        <external_bootstrap>true</external_bootstrap>
        <avail_trigger>all</avail_trigger>
        <ad_markers>none</ad_markers>
        <broadcast_time>false</broadcast_time>
        <ignore_web_delivery_allowed>false</ignore_web_delivery_allowed>
        <ignore_no_regional_blackout>false</ignore_no_regional_blackout>
        <absolute_timestamps>false</absolute_timestamps>
      </filter_settings>
    </filter>
  </filters>
</content>
```

Response

The response repeats back the filter that you created, with the addition of the following tags:

- `id`: Unique IDs for the filter and each sub-element as described in this section.

- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI, as follows:

  ```
  http://<Delta IP address>:8080/out/i/<filter ID>.mpd
  ```

  where `filter ID` is the ID for this filter, assigned once the filter is created.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if `output_uri` contains a value. Contains the custom endpoint URI, as follows:

  ```
  http://<Delta IP address:8080>/out/u/<output_url>.mpd
  ```

This example response is a representation. The `default_endpoint` now shows a value. The `filter_settings` shows the default values. The `filter`, `filter_settings`, `stream_set`, `video_track`, and `audio_track` elements are all assigned unique IDs. Your results may vary.

```
<?xml version="1.0" encoding="UTF-8"?>
<contents href="/contents" product="Delta" version="2.3.0.123456">
  <content href="/contents/39" product="Delta" version="2.3.0.123456>
    <filters href="/contents/39/filters" product="Delta" version="2.3.0.123456">
      <filter href="/contents/39/filters/18" product="Delta" version="2.3.0.123456">
        <id>18</id>
        <parent_id/>
        <endpoint>true</endpoint>
        <default_endpoint_uri>http://10.24.34.2/out/i/18.f4m</default_endpoint_uri>
        <custom_endpoint_uri>http://10.24.34.2/out/u/movie21.f4m</
custom_endpoint_uri><output_url>movie21</output_url>
        <filter_type>hds_package</filter_type>
        <filter_settings>
          <id>1</id>
          <fragment_duration>2</fragment_duration>
          <index_duration>60</index_duration>
          <external_bootstrap>true</external_bootstrap>
          <avail_trigger>all</avail_trigger>
          <ad_markers>none</ad_markers>
          <broadcast_time>false</broadcast_time>
          <ignore_web_delivery_allowed>false</ignore_web_delivery_allowed>
          <ignore_no_regional_blackout>false</ignore_no_regional_blackout>
          <absolute_timestamps>false</absolute_timestamps>
        </filter_settings>
        <stream_sets>
          <stream_set>
```

```
            <id>6</id>
            <stream_index>0</stream_index>
            <iframe_only>false</iframe_only>
            <video_tracks>
              <video_track>
                <id>13</id>
                <track_index>0</track_index>
                <track_id>54</track_id>
              </video_track>
            </video_tracks>
            <audio_tracks>
              <audio_track>
                <id>14</id>
                <track_index>1</track_index>
                <track_id>56</track_id>
              </audio_track>
            </audio_tracks>
          </stream_set>
          <stream_set>
            <id>7</id>
            <stream_index>1</stream_index>
            <iframe_only>false</iframe_only>
            <video_tracks>
              <video_track>
                <id>15</id>
                <track_index>0</track_index>
                <track_id>53</track_id>
               </video_track>
            </video_tracks>
            <audio_tracks>
              <audio_track>
                <id>16</id>
                <track_index>1</track_index>
                <track_id>56</track_id>
              </audio_track>
            </audio_tracks>
          </stream_set>
        </stream_sets>
      </filter>
    </filters>
  </content>
</contents>
```

Topics

- The filter Element
- The filter_settings Element
- The stream_set Element

The `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint.

Tag	Type	Description
		The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only. The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. Delta automatically creates this customized address, including the `output_url` . Delta stores the address in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.

Tag	Type	Description
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).

Tag	Type	Description
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
`fragment_duration`	integer	The length of fragments (in seconds) that Delta generates. Delta rounds actual fragments to the nearest multiple of the source content fragment length. The default is the value from the ingested content.
`index_duration`	integer	The length of the live manifest (in seconds). Delta ignores this parameter for VOD content.
`external_bootstrap`	boolean	This tag specifies what Delta does with the bootstrap file. • **`true`** is the default and means that Delta emits the bootstrap as a separate file. Some HDS players require external bootstrap files. • **`false`** means that Delta embeds the bootstrap in the manifest file.
`avail_trigger`	string	This tag specifies which type of ad avail SCTE-35 messages result in Delta including

Tag	Type	Description
		`cueInfo` tags in the output manifest. Typically, you should select the trigger to match the ad avail type that the input contains. • **`all`** is the default. The input uses splice inserts to identify ad avails. The input might also contain messages for other events such as chapters or programs. These events also result in `cueInfo` tags in the output manifest. • **`placement_opportunity`** indicates that the input uses time signals with a segmentation type of `placement opportunity`. The input might also contain messages for other events such as chapters or programs. These events also result in `cueInfo` tags in the output manifest.

Tag	Type	Description
`ad_markers`	string	This tag specifies how ad markers appear in the output. • **`none`** is the default and means that ad markers are not included in the output. • **`scte35_enhanced`** means that Delta generates ad markers and blackout tags based on SCTE-35 messages in the UDP or HLS input. • **`passthrough`** means that the output manifest contains a copy of the SCTE-35 ad markers from the input HLS manifest.
`broadcast_time`	boolean	This tag specifies how ID3 timed metadata messages appear in the output manifest. • **`false`** is the default. Delta does not insert messages in the output manifest. • **`true`** means that Delta generates messages every 5 seconds and inserts them in the output manifest with the content ingest time.

Tag	Type	Description
`ignore_web_delivery_allowed`	boolean	This tag specifies how Delta handles `web_delivery_allowed` tags from the input manifest. Applicable when `ad_markers` is **`scte35_enhanced`** only. • **`false`** is the default. SCTE-35 segmentation descriptors with `web_delivery_allowed` set to **`0`** trigger blackouts or avails. • **`true`** means that blackouts and avails are *not* triggered when `web_delivery_allowed` is set to **`0`**.

Tag	Type	Description
`ignore_no_regional_blackout`	boolean	This tag specifies how Delta handles `no_regional_blackout` tags in the input manifest. Applies when `ad_markers` is **`scte35_enhanced`** only. • **`false`** is the default. SCTE-35 segmentation descriptors with `no_regional_blackout` set to **`0`** trigger blackouts or avails. • **`true`** means that blackouts and avails are *not* triggered when `no_regional_blackout` is set to **`0`**.
`absolute_timestamps`	boolean	The timestamp always specifies the time (in milliseconds) that Delta ingests the corresponding segment, relative to the base time. This tag defines the base time. • When **`true`**, the base time is the epoch (00:00:00 UTC, Thursday, 1 January 1970). • When **`false`**, the base time is the start of the start of the stream.

The `stream_set` Element

The `stream_set` element holds information about how Delta handles the audio, video, and subtitles in a stream. The following table describes the settings for the `stream_set` element.

Tag	Type	Description
`stream_index`	integer	An optional number entered in each stream set in the filter to sort the sets in relation to each other. A lower number puts the set higher in the list. Numbers can be skipped. For example, if you later remove a stream set from the filter, there is no need to renumber the other sets.
`audio_tracks`	object	This tag contains informati on about the audio tracks to include in the stream set. To specify more than one audio track in the stream set, include multiple `audio_tra ck` objects. See the table below for the tags to enter in the `audio_track` .
`video_tracks`	object	This tag contains informati on about the video tracks to include in the stream set. To specify more than one video track in the stream set, include multiple `video_tra ck` objects. See the table

Tag	Type	Description
		below for the tags to enter in the `video_track` .
`subtitle_tracks`	object	This tag contains information about the subtitle or captions tracks to include in the stream set. To specify more than one subtitle or captions track in the stream set, include multiple `subtitle_track` objects. See the table below for the tags to enter in the `subtitle_track` .

The `video_track`, `audio_track`, and `subtitle_track` Elements

The following table describes the settings for the `video_track`, `audio_track`, and `subtitle_track` elements.

Tag	Type	Description
`track_id`	integer	Read-only. The Delta-assigned numeric value for the track.
`track_index`	integer	A number used to sort the tracks in relation to each other if the stream set contains more than one track of the same type (video, audio, or subtitle). Enter a number in each *xx* track. A lower number puts the track higher in the list.

Tag	Type	Description
		Numbers can be skipped. For example, if you later remove a track from the set, there is no need to renumber the other tracks.

PUT: Create an HLS Package Output Filter

To create an Apple HTTP Live Streaming (HLS) package output filter on a content object, send a PUT request. The following sections describe how to format the request and provide examples of the request and response.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID
```

Body of HTTP

The body of your request is XML content consisting of one `content` element that holds:

- One `filters` container and one `filter` element that holds several tags, and also holds:
  - One `filter_settings` element that holds several tags.
  - One `stream_sets` container and zero or one `stream_set` elements (zero only if `use_default_stream_sets` is **`true`**). The `stream_set` element is required for DVB-Text and DVB-Sub subtitles.

    If present, the stream_set element holds several tags and also holds:

    - One `video_tracks` container and zero or one `video_track` element that holds several tags.
    - One `audio_tracks` container and zero or one `audio_track` element that holds several tags.
    - One `subtitle_tracks` container and zero or one `subtitle_track` element that holds several tags.

Request Example

This request modifies the existing content entity that has ID 39 to add an HLS output filter. The request omits the `parent_id`, which means it is a top-level filter. It includes an `output_url` (to create a custom endpoint that contains `movie21`).

It includes three regular stream sets, each of which contains one (different) video and one default audio (which is the same track in each). It also includes one rendition group that contains alternative audio tracks. Finally, it includes one I-frame stream set.

```
PUT http://10.24.34.2:8080/contents/39
---------------------------------------
<content>
  <filters>
    <filter>
      <endpoint>false</endpoint><output_url>movie21</output_url>
      <filter_type>hls_package</filter_type>
      <filter_settings>
        <segment_duration>2</segment_duration>
        <index_duration>60</index_duration>
        <playlist_type/>
        <avail_trigger>all</avail_trigger>
        <ad_markers>none</ad_markers>
        <broadcast_time>false</broadcast_time>
        <ignore_web_delivery_allowed>false</ignore_web_delivery_allowed>
        <ignore_no_regional_blackout>false</ignore_no_regional_blackout>
        <enable_blackout>false</enable_blackout>
        <enable_network_end_blackout>false</enable_network_end_blackout>
        <network_id/>
      </filter_settings>
      <stream_sets>
        <stream_set>
          <stream_index>0</stream_index>
          <video_tracks>
            <video_track>
              <track_id>54</track_id>
            </video_track>
          </video_tracks>
          <audio_tracks>
            <audio_track>
            <track_id>56</track_id>
          </audio_track>
        </audio_tracks>
```

```
        </stream_set>
        <stream_set>
          <stream_index>1</stream_index>
            <video_tracks>
              <video_track>
                <track_id>53</track_id>
              </video_track>
            </video_tracks>
            <audio_tracks>
              <audio_track>
              <track_id>56</track_id>
            </audio_track>
          </audio_tracks>
        </stream_set>
        <stream_set>
          <stream_index>2</stream_index>
            <video_tracks>
              <video_track>
                <track_id>55</track_id>
              </video_track>
            </video_tracks>
            <audio_tracks>
              <audio_track>
                <track_id>56</track_id>
              </audio_track>
            </audio_tracks>
          </stream_set>
          <stream_set>
            <stream_index>3</stream_index>
            <rendition_group_name>Audio</rendition_group_name>
            <audio_tracks>
              <audio_track>
                <track_index>0</track_index>
                <rendition_group_value>ES</rendition_group_value>
                <track_id>59</track_id>
              </audio_track>
              <audio_track>
                <track_index>1</track_index>
                <rendition_group_value>FR</rendition_group_value>
                <track_id>60</track_id>
              </audio_track>
              <audio_track>
                <track_index>2</track_index>
                <rendition_group_value>DE</rendition_group_value>
```

```
              <track_id>61</track_id>
            </audio_track>
          </audio_tracks>
          <subtitle_tracks/>
        </stream_set>
        <stream_set>
          <stream_index>4</stream_index
          <iframe_only>true</iframe_only>
          <video_tracks>
            <video_track>
              <track_id>54</track_id>
            </video_track>
          </video_tracks>
        </stream_set>
      </stream_sets>
    </filter>
  </filters>
</content>
```

Response

The response repeats back the filter that you created, with the addition of the following tags:

- `id`: Unique IDs for the filter and each sub-element as described in this section.
- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI, as follows:

  ```
  http://<Delta IP address>:8080/out/i/<filter ID>.m3u8
  ```

  where `filter ID` is the ID for this filter, assigned once the filter is created.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if `output_uri` contains a value. Contains the custom endpoint URI, as follows:

  ```
  http://<Delta IP address:8080>/out/u/<output_url>.m3u8
  ```

This example response is a representation. The `default_endpoint` now shows a value. The `filter_settings` shows the default values. The `filter`, `filter_settings`, `stream_set`, `video_track`, and `audio_track` elements are all assigned unique IDs. Your results may vary.

```
<?xml version="1.0" encoding="UTF-8"?>
<contents href="/contents" product="Delta" version="2.3.0.123456">
```

```
  <content href="/contents/39" product="Delta" version="2.3.0.123456">
    <id>39</id>
    <name>weather_usa.isml</name>
    <path></path>
    <type>hls</type>
    <filters href="/contents/39/filters" product="Delta" version="2.3.0.123456">
      <filter href="/contents/39/filters/20" product="Delta" version="2.3.0.123456">
        <id>20</id>
        <parent_id/>
        <default_endpoint_uri>http://10.24.34.2/out/i/20.m3u8</default_endpoint_uri>
        <custom_endpoint_uri>http://10.24.34.2/out/u/movie21.m3u8</
custom_endpoint_uri><output_url>movie21</output_url>
        <filter_settings>
          <id>2</id>
          <segment_duration>2</segment_duration>
          <index_duration>60</index_duration>
          <playlist_type/>
          <avail_trigger>all</avail_trigger>
          <ad_markers>none</ad_markers>
          <broadcast_time>false</broadcast_time>
          <ignore_web_delivery_allowed>false</ignore_web_delivery_allowed>
          <ignore_no_regional_blackout>false</ignore_no_regional_blackout>
          <enable_blackout>false</enable_blackout>
          <enable_network_end_blackout>false</enable_network_end_blackout>
          <network_id/>
        </filter_settings>
        <stream_sets>
          <stream_set>
            <id>12</id>
            <stream_index>0</stream_index>
            <iframe_only>false</iframe_only>
            <video_tracks>
              <video_track>
              <id>25</id>
              <track_index>0</track_index>
              <track_id>54</track_id>
            </video_track>
          </video_tracks>
          <audio_tracks>
            <audio_track>
              <id>26</id>
              <track_index>1</track_index>
   <track_id>56</track_id>
   </audio_track>
```

```
</audio_tracks>
<subtitle_tracks />
</stream_set>
<stream_set>
<id>13</id>
<stream_index>1</stream_index>
<iframe_only>false</iframe_only>
<video_tracks>
<video_track>
<id>28</id>
<track_index>0</track_index>
<track_id>53</track_id>
</video_track>
</video_tracks>
<audio_tracks>
<audio_track>
<id>29</id>
<track_index>1</track_index>
<track_id>56</track_id>
</audio_track>
<subtitle_tracks />
</stream_set>
<stream_set>
<id>14</id>
<stream_index>2</stream_index>
<iframe_only>false</iframe_only>
<video_tracks>
<video_track>
<id>30</id>
<track_index>0</track_index>
<track_id>55</track_id>
</video_track>
</video_tracks>
<audio_tracks>
<audio_track>
<id>31</id>
<track_index>1</track_index>
<track_id>56</track_id>
</audio_track>
</audio_tracks>
<subtitle_tracks />
</stream_set>
<stream_set>
<id>15</id>
```

```
    <stream_index>4</stream_index>
    <rendition_group_name>Audio</rendition_group_name>
    <iframe_only>false</iframe_only>
    <audio_tracks>
    <audio_track>
    <id>27</id>
    <track_index>0</track_index>
    <rendition_group_value>ES</rendition_group_value>
    <track_id>59</track_id>
    </audio_track>
    <audio_track>
    <id>27</id>
    <track_index>1</track_index>
    <rendition_group_value>FR</rendition_group_value>
    <track_id>60</track_id>
    </audio_track>
    <audio_track>
    <id>27</id>
    <track_index>2</track_index>
    <rendition_group_value>DE</rendition_group_value>
    <track_id>61</track_id>
    <audio_track>
    </audio_tracks>
    <subtitle_tracks />
    </stream_set>
    <stream_set>
    <id>16</id>
    <stream_index>5</stream_index>
    <iframe_only>true</iframe_only>
             <video_tracks>
               <video_track>
                 <id>32</id>
                 <track_index>0</track_index>
                 <track_id>54</track_id>
               </video_track>
             </video_tracks>
             <subtitle_tracks />
           </stream_set>
         </stream_sets>
       </filter>
     </filters>
   </content>
 </contents>
```

Topics

- The filter Element
- The filter_settings Element
- The stream_set Element

The `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint.

Tag	Type	Description
		The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only. The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. Delta automatically creates this customized address, including the `output_url` . Delta stores the address in the `custom_endpoint_uri` tag. **Warning** If you have HLS and CMAF output filters on the same content, then you must make sure that the `output_url` value is unique. Otherwise, both filters will have the same URL and the endpoints will collide.

Tag	Type	Description
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.

Tag	Type	Description
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
`segment_duration`	integer	The length of TS segments that Delta generates. Delta rounds actual segments to hit the next available

Tag	Type	Description
		segmentation point in the source content. The default is the value from the ingested content.
`index_duration`	integer	The length of the live manifest (in seconds). Delta ignores this parameter for VOD content.
`playlist_type`	string	This tag specifies how Delta handles the `EXT-X-PLAYLIST-TYPE` tag in the output manifest for VOD and live-to-VOD content. • Empty (no value) is the default. Delta doesn't insert `EXT-X-PLAYLIST_TYPE` tags in the output. • **`EVENT`** means that Delta inserts the `EXT-X-PLAYLIST-TYPE:EVENT` in the manifest. • **`VOD`** means that Delta inserts the `EXT-X-PLAYLIST-TYPE:VOD` tag in the manifest.

Tag	Type	Description
`avail_trigger`	string	This tag specifies which type of ad avail SCTE-35 messages result in Delta including `cueInfo` tags in the output manifest. Typically, you should select the trigger to match the ad avail type that the input contains. • **`all`** is the default. The input uses splice inserts to identify ad avails. The input might also contain messages for other events such as chapters or programs. These events also result in `cueInfo` tags in the output manifest. • **`placement_opportunity`** indicates that the input uses time signals with a segmentation type of `placement opportunity`. The input might also contain messages for other events such as chapters or programs. These events also result in `cueInfo` tags in the output manifest.

Tag	Type	Description
`ad_markers`	string	This tag specifies how ad markers appear in the output. • **`none`** is the default and means that ad markers are not included in the output. • **`scte35_enhanced`** means that Delta generates ad markers and blackout tags based on SCTE-35 messages in the UDP or HLS input. • **`passthrough`** means that the output manifest contains a copy of the SCTE-35 ad markers from the input HLS manifest.
`broadcast_time`	boolean	This tag specifies how ID3 timed metadata messages appear in the output manifest. • **`false`** is the default. Delta does not insert messages in the output manifest. • **`true`** means that Delta generates messages every 5 seconds and inserts them in the output manifest with the content ingest time.

Tag	Type	Description
`ignore_web_delivery_allowed`	boolean	This tag specifies how Delta handles `web_delivery_allowed` tags from the input manifest. Applicable when `ad_markers` is **`scte35_enhanced`** only. • **`false`** is the default. SCTE-35 segmentation descriptors with `web_delivery_allowed` set to **`0`** trigger blackouts or avails. • **`true`** means that blackouts and avails are *not* triggered when `web_delivery_allowed` is set to **`0`**.

Tag	Type	Description
`ignore_no_regional_blackout`	boolean	This tag specifies how Delta handles `no_regional_blackout` tags in the input manifest. Applies when `ad_markers` is **`scte35_enhanced`** only. • **`false`** is the default. SCTE-35 segmentation descriptors with `no_regional_blackout` set to **`0`** trigger blackouts or avails. • **`true`** means that blackouts and avails are *not* triggered when `no_regional_blackout` is set to **`0`**.

Tag	Type	Description
`enable_blackout`	boolean	This tag specifies how `blackout` tags appear in the output manifest. Applicable when `ad_markers` is **`scte35_enhanced`** only. • **`false`** is the default and means that Delta does not insert `blackout` tags into the output manifest. • **`true`** means that Delta adds `blackout` tags to the output manifest based on SCTE-35 `Program`, `Chapter`, and `Not Indicated` segmentation descriptors when `web_delivery_allowed` or `no_regional_blackout` is set to **`0`**.

Tag	Type	Description
`enable_network_end_blackout`	boolean	This tag specifies how `blackout` tags appear in the output manifest. Applicable when `ad_markers` is **`scte35_enhanced`** only. • **`false`** is the default and means that Delta does not insert `blackout` tags into the output manifest. • **`true`** means that Delta adds `blackout` tags to the output manifest based on SCTE-35 `Network Start/End` segmentation descriptors. When **`true`**, you must also provide a value for `network_id` .
`network_id`	string	This tag specifies the EIDR ID of the network in the format 10.nnnn/xxxx-xxxx-xxxx-xxxx-xxxx-c (not case sensitive). Only network end events with this ID trigger blackout. Required when `enable_network_end_blackout` is **`true`**.

Tag	Type	Description
`include_program_date_time`	boolean	This tag specifies how Delta handles `EXT-X-PROGRAM-DATE-TIME` in the output manifest. • **`false`** is the default and means that Delta does not insert `EXT-X-PROGRAM-DATE-TIME` tags in the output manifest. • **`true`** means that Delta does insert `EXT-X-PROGRAM-DATE-TIME` tags in the output manifest, based on the input manifest. If the source content is HLS and it contains `EXT-X-PROGRAM-DATE-TIME` tags, then Delta uses the source time for the value. Otherwise, Delta uses the time from when the input filter processed the source segment.
`program_date_time_interval`	integer	When `include_program_date_time` is **`true`**, this tag specifies the frequency (in seconds) at which Delta inserts `EXT-X-PROGRAM-DATE-TIME` tags in the output manifest. The default is **`600`**.

The `stream_set` Element

The `stream_set` element holds information about how Delta handles the audio, video, and subtitles in a stream. The following table describes the settings for the `stream_set` element.

Tag	Type	Description
`stream_index`	integer	An optional number entered in each stream set in the filter to sort the sets in relation to each other. A lower number puts the set higher in the list. Numbers can be skipped. For example, if you later remove a stream set from the filter, there is no need to renumber the other sets.
`audio_tracks`	object	This tag contains informati on about the audio tracks to include in the stream set. To specify more than one audio track in the stream set, include multiple `audio_tra ck` objects. See the table below for the tags to enter in the `audio_track` .
`video_tracks`	object	This tag contains informati on about the video tracks to include in the stream set. To specify more than one video track in the stream set, include multiple `video_tra ck` objects. See the table

Tag	Type	Description
		below for the tags to enter in the `video_track` .
`subtitle_tracks`	object	This tag contains information about the subtitle or captions tracks to include in the stream set. To specify more than one subtitle or captions track in the stream set, include multiple `subtitle_track` objects. See the table below for the tags to enter in the `subtitle_track` .
`rendition_group_name`	string	A name entered to set this stream set up as a rendition group: all audio tracks are together in a rendition group. The video track works with any audio track in the rendition group. Leave blank to use a regular stream set.

Tag	Type	Description
`iframe_only`	boolean	This setting applies to stream sets with a single video track. The value is ignored in all other situations. • **`true`** means that the stream set will include an additional `EXT-X-I-FRAMES` tag. This tag prompts Delta to include an I-frame-only stream in the stream set with the other tracks that you specify. • **`false`** is the default and means that an I-frame-only stream is not included in the stream set.
`announced_bitrate`	boolean	Bitrate for the stream set. This value is used in adaptive bitrate bitrate manifest creation. If left blank, Delta auto-detects the value.

Tag	Type	Description
`audio_group_name`	string	This tag specifies what audio rendition group the video track uses. Options are: • Omit the tag: the default audio rendition group (the first one in the playlist) is used. • Include the tag with a rendition group name: the indicated audio rendition group is used. • Include the tag without a value (`audio_group_name/`): no audio rendition group is used. This is commonly used if the video track already has audio muxed in.

The `video_track`, `audio_track`, and `subtitle_track` Elements

The following table describes the settings for the `video_track`, `audio_track`, and `subtitle_track` elements.

Tag	Type	Description
`track_id`	integer	Read-only. The Delta-assigned numeric value for the track.
`track_index`	integer	A number used to sort the tracks in relation to each other if the stream set

Tag	Type	Description
		contains more than one track of the same type (video, audio, or subtitle). Enter a number in each *xx* track. A lower number puts the track higher in the list. Numbers can be skipped. For example, if you later remove a track from the set, there is no need to renumber the other tracks.
`rendition_group_value`	string	For audio and subtitle tracks only. A name assigned to each track if the stream set is a rendition group (you have defined a value for `rendition_group_name`). For example, assign a language designator.
`teletext_page_number`	hexadecimal string	Used on the `subtitle_track` only. The teletext page number within the incoming stream from which captions are extracted. Must be a three-digit hexadecimal string between the range of 100 and 8FF.

Tag	Type	Description
`language`	string	The language of the incoming track. Used on the `subtitle_track` only.

PUT: Create an MP4 Package Output Filter

To create an MP4 package output filter on a content object, send a PUT request. The following sections describe how to format the request and provide examples of the request and response.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID
```

Body of HTTP

The body of your request is XML content consisting of one `content` element that holds:

- One `filters` container and one `filter` element that holds several tags, and also holds:
  - One `filter_settings` element that holds several tags.
  - One `stream_sets` container and zero or more `stream_set` elements (zero only if `use_default_stream_sets` is true). If present, each `stream_set` element holds several tags and also holds:
    - One `video_tracks` container and one `video_track` element that holds several tags.
    - One `audio_tracks` container and zero or one `audio_track` element that holds several tags.

Request Example

This request modifies the existing content entity that has ID 39 to add an MP4 output filter. The request includes a `parent_id` because it is not a top-level filter. It includes an `output_url` (to create a custom endpoint that contains `movie21`). It includes two stream sets, each of which holds one (different)video and one default audio (which is the same track in each).

```
PUT http://10.24.34.2:8080/conents/39
---------------------------------------------
```

```
<content>
  <filters>
    <filter>
      <parent_id>21</parent_id>
      <endpoint>true</endpoint><output_url>movie21</output_url>
      <filter_type>mp4_package</filter_type>
      <filter_settings>
        <major_brand/>
        <include_cslg>false</include_cslg>
      </filter_settings>
      <stream_sets>
        <stream_set>
          <stream_index>0</stream_index>
          <video_tracks>
            <video_track>
            <track_id>54</track_id>
            </video_track>
          </video_tracks>
          <audio_tracks>
            <audio_track>
              <track_id>56</track_id>
            </audio_track>
          </audio_tracks>
        </stream_set>
        <stream_set>
          <stream_index>1</stream_index>
          <video_tracks>
              <video_track>
                <track_id>53</track_id>
              </video_track>
          </video_tracks>
          <audio_tracks>
            <audio_track>
              <track_id>56</track_id>
            </audio_track>
          </audio_tracks>
        </stream_set>
      </stream_sets>
    </filter>
  </filters>
</content>
```

Response

The response repeats back the filter that you created, with the addition of the following tags:

- `id`: Unique IDs for the filter and each sub-element as described in this section.
- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI, as follows:

  ```
  http://<Delta IP address>:8080/out/i/<filter ID>.mp4
  ```

  where `filter ID` is the ID for this filter, assigned once the filter is created.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if `output_uri` contains a value. Contains the custom endpoint URI, as follows:

  ```
  http://<Delta IP address:8080>/out/u/<output_url>.mp4
  ```

This example response is a representation. The `default_endpoint` now shows a value. The `filter_settings` shows the default values. The `filter`, `filter_settings`, `stream_set`, `video_track`, and `audio_track` elements are all assigned unique IDs. Your results may vary.

```
<?xml version="1.0" encoding="UTF-8"?>
  <contents href="/contents" product="Delta" version="2.3.0.123456">
   <content href="/contents/39" product="Delta" version="2.3.0.123456">
   <filters href="/contents/39/filters" product="Delta" version="2.3.0.123456">
   <filter href="/contents/39/filters/30" product="Delta" version="2.3.0.123456">
   <id>30</id>
   <parent_id>21</parent_id>
   <endpoint>true</endpoint>
   <default_endpoint_uri>http://10.24.34.2/out/i/30.mp4</default_endpoint_uri>
   <custom_endpoint_uri>http://10.24.34.2/out/u/movie21.mp4</
custom_endpoint_uri><output_url>movie21</output_url>
   <filter_type>mp4_package</filter_type>
   <filter_settings>
   <id>1</id>
   <major_brand/>
   <include_cslg>false</include_cslg>
   </filter_settings>
   <stream_sets>
   <stream_set>
   <id>28</id>
   <stream_index>0</stream_index>
```

```
  <iframe_only>false</iframe_only>
  <video_tracks>
  <video_track>
  <id>47</id>
  <track_index>0</track_index>
  <track_id>54</track_id>
  </video_track>
  </video_tracks>
  <audio_tracks>
  <audio_track>
  <id>48</id>
  <track_index>1</track_index>
  <track_id>56</track_id>
  </audio_track>
  </audio_tracks>
  </stream_set>
  <stream_set>
  <id>29</id>
  <stream_index>1</stream_index>
  <iframe_only>false</iframe_only>
  <video_tracks>
  <video_track>
  <id>49</id>
  <track_index>0</track_index>
  <track_id>53</track_id>
  </video_track>
  </video_tracks>
  <audio_tracks>
  <audio_track>
  <id>50</id>
  <track_index>1</track_index>
  <track_id>56</track_id>
  </audio_track>
 </audio_tracks>
  </stream_set>
  </stream_sets>
  </filter>
  </filters>
  </content>
 </contents>
```


Topics

- The filter Element
- The filter_settings Element
- The stream_set Element

The `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.

Tag	Type	Description
`default_endpoint_uri`	string	Read-only. The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. Delta automatically creates this customized address, including the `output_url` . Delta stores the address in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.

Tag	Type	Description
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).

Tag	Type	Description
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description

`include_cslg`	boolean	This tag informs how the Composition Time Stamp (CTS) is configured.
		• **`false`** is the default. Delta creates an `.iso` MPEG-4-compliant stream.
		• **`true`** configures the CTS in the following ways:
		  • The file composition times start at zero.
		  • Composition times in the `composition time to sample` (ctts) box for B-frames is negative.
		  • A `composition shift least greatest` (cslg) box is included per 14496-1 amendment 1.
		  These settings improve compatibility with Apple players and tools.

`major_brand`	string	When provided, this 4-character value overrides the `Major Brand` field in the output file.

The `stream_set` Element

The `stream_set` element holds information about how Delta handles the audio, video, and subtitles in a stream. The following table describes the settings for the `stream_set` element.

Tag	Type	Description
`stream_index`	integer	An optional number entered in each stream set in the filter to sort the sets in relation to each other. A lower number puts the set higher in the list. Numbers can be skipped. For example, if you later remove a stream set from the filter, there is no need to renumber the other sets.
`audio_tracks`	object	This tag contains informati on about the audio tracks to include in the stream set. To specify more than one audio track in the stream set, include multiple `audio_tra ck` objects. See the table below for the tags to enter in the `audio_track` .
`video_tracks`	object	This tag contains informati on about the video tracks to include in the stream set. To specify more than one video track in the stream set, include multiple `video_tra ck` objects. See the table

Tag	Type	Description
		below for the tags to enter in the `video_track` .
`subtitle_tracks`	object	This tag contains information about the subtitle or captions tracks to include in the stream set. To specify more than one subtitle or captions track in the stream set, include multiple `subtitle_track` objects. See the table below for the tags to enter in the `subtitle_track` .

The `video_track`, `audio_track`, and `subtitle_track` Elements

The following table describes the settings for the `video_track`, `audio_track`, and `subtitle_track` elements.

Tag	Type	Description
`track_id`	integer	Read-only. The Delta-assigned numeric value for the track.
`track_index`	integer	A number used to sort the tracks in relation to each other if the stream set contains more than one track of the same type (video, audio, or subtitle). Enter a number in each *xx* track. A lower number puts the track higher in the list.

Tag	Type	Description
		Numbers can be skipped. For example, if you later remove a track from the set, there is no need to renumber the other tracks.

PUT: Create an MPEG-TS Package Output Filter

To create an MPEG transport stream (MPEG-TS) package output filter on a content object, send a PUT request. The following sections describe how to format the request and provide examples of the request and response.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID
```

Body of HTTP

The body of your request is XML content consisting of one `content` element that holds:

- One `filters` container and one `filter` element that holds several tags, and also holds:
  - One `filter_settings` element that holds several tags.
  - One `stream_sets` container and zero or more `stream_set` elements (zero only if `use_default_stream_sets` is **`true`**). If present, each `stream_set` element holds several tags and also holds:
    - One `video_tracks` container and zero or more `video_track` elements that hold several tags.
    - One `audio_tracks` container and zero or more `audio_track` elements that hold several tags.

Request Example

This request modifies the existing content entity that has ID 39 to add an MP4 output filter. The request includes a `parent_id` because it is not a top-level filter. It includes an `output_url` (to create a custom endpoint that contains `movie21`). It includes two stream sets, each of which contains one (different) video and one default audio (which is the same track in each).

```
PUT http://10.24.34.2:8080/contents/39
---------------------------------------
<content>
  <filters>
    <filter>
      <parent_id>21</parent_id>
      <endpoint>true</endpoint>
      <output_url>movie21</output_url>
      <filter_type>mpeg_ts_package</filter_type>
      <filter_settings>
        <bitrate/>
        <vbr>false</vbr>
        <dvb>false</dvb>
        <audio_packets_per_pes>2</audio_packets_per_pes>
        <pcr_every_pes>true</pcr_every_pes>
        <pcr_period/>
        <pat_interval>100</pat_interval>
        <pmt_interval>100</pmt_interval>
      </filter_settings>
      <stream_sets>
        <stream_set>
          <stream_index>0</stream_index>
          <video_tracks>
            <video_track>
              <track_id>54</track_id>
            </video_track>
          </video_tracks>
          <audio_tracks>
            <audio_track>
              <track_id>56</track_id>
            </audio_track>
          </audio_tracks>
        </stream_set>
        <stream_set>
        <stream_index>1</stream_index>
          <video_tracks>
            <video_track>
              <track_id>53</track_id>
            </video_track>
          </video_tracks>
          <audio_tracks>
            <audio_track>
              <track_id>56</track_id>
```

```
            </audio_track>
          </audio_tracks>
        </stream_set>
      </stream_sets>
    </filter>
  </filters>
 </content>
```

Response

The response repeats back the filter that you created, with the addition of the following tags:

- `id`: Unique IDs for the filter and each sub-element as described in this section.
- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI, as follows:

  ```
  http://<Delta IP address>:8080/out/i/<filter ID>.ts
  ```

  where `filter ID` is the ID for this filter, assigned once the filter is created.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if `output_uri` contains a value. Contains the custom endpoint URI, as follows:

  ```
  http://<Delta IP address:8080>/out/u/<output_url>.ts
  ```

This example response is a representation. The `default_endpoint` now shows a value. The `filter_settings` shows the default values. The `filter`, `filter_settings`, `stream_set`, `video_track`, and `audio_track` elements are all assigned unique IDs. Your results may vary.

```
<?xml version="1.0" encoding="UTF-8"?>
<contents href="/contents" product="Delta" version="2.3.0.123456">
  <content href="/contents/39" product="Delta" version="2.3.0.123456">
    <filters href="/contents/39/filters" product="Delta" version="2.3.0.123456">
      <filter href="/contents/39/filters/31" product="Delta" version="2.3.0.123456">
        <id>31</id>
        <parent_id>21</parent_id>
        <endpoint>true</endpoint>
        <default_endpoint_uri>http://10.4.142.112/out/i/31.ts</default_endpoint_uri>
        <custom_endpoint_uri>http://10.4.142.112/out/u/movie21.ts</
custom_endpoint_uri>
        <output_url>movie21</output_url>
        <filter_type>mpeg_ts_package</filter_type>
```

```
        <filter_settings>
          <id>1</id>
          <bitrate/>
          <vbr>false</vbr>
          <dvb>false</dvb>
          <audio_packets_per_pes>2</audio_packets_per_pes>
          <pcr_every_pes>true</pcr_every_pes>
          <pcr_period/>
          <pat_interval>100</pat_interval>
          <pmt_interval>100</pmt_interval>
        </filter_settings>
        <stream_sets>
          <stream_set>
            <id>31</id>
            <stream_index>0</stream_index>
            <iframe_only>false</iframe_only>
            <video_tracks>
              <video_track>
                <id>53</id>
                <track_index>0</track_index>
                <track_id>54</track_id>
              </video_track>
            </video_tracks>
            <audio_tracks>
              <audio_track>
                <id>54</id>
                <track_index>1</track_index>
                <track_id>56</track_id>
              </audio_track>
            </audio_tracks>
          </stream_set>
          <stream_set>
            <id>32</id>
            <stream_index>1</stream_index>
            <iframe_only>false</iframe_only>
            <video_tracks>
                <video_track>
                  <id>55</id>
                  <track_index>0</track_index>
                  <track_id>53</track_id>
                </video_track>
            </video_tracks>
            <audio_tracks>
              <audio_track>
```

```
                <id>56</id>
                <track_index>1</track_index>
                <track_id>56</track_id>
              </audio_track>
            </audio_tracks>
          </stream_set>
        </stream_sets>
      </filter>
    </filters>
  </content>
</contents>
```

Topics

- The filter Element
- The filter_settingsElement
- The stream_set Element

The `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag.

Tag	Type	Description
		To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only. The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. Delta automatically creates this customized address, including the `output_url` . Delta stores the address in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .

Tag	Type	Description
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).

Tag	Type	Description
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The `filter_settings`Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
`bitrate`	integer	The output bitrate of the transport stream in bits per second. The default value is **`NULL`** and lets the muxer automatically determine the approrpriate bitrate. Other common values are **`3750000`**, **`7500000`**, and **`15000000`**.
`vbr`	boolean	This tag specifies how Delta handles the `bitrate` value. • **`false`** is the default. Delta provides a constant bitrate by inserting `NULL` packets into the stream to fill the specified `bitrate` value. • **`true`** means that the `bitrate` value acts as the maximum bitrate. Delta does not insert `NULL` packets into the transport stream.
dvb	boolean	This tag specifies the audio buffer model. • **`false`** is the default. Delta uses the Advanced Televisio n Systems Committee

Tag	Type	Description
		(ATSC) standards for buffering. • **`true`** means that Delta uses the DVB buffer model for Dolby Digital audio.
`audio_packets_per_pes_bitrate`	integer	The number of audio packets that Delta inserts for each PES packet. Default is **2**.
`pcr_every_pes`	boolean	This tag specifies how Delta handles the `program clock reference` (PCR) in the output manifest. • **`true`** is the default. Delta inserts a PCR value for every `packetized elementary stream` (PES) header. This parameter is effective only when the PCR program identifier (PID) is the same as the video or audio elementary system. • **`false`** means that Delta inserts a PCR value based on the value of the `pcr_period` tag.
`pcr_period`	integer	Nominal time in milliseconds between `program clock references` (PCRs) in the transport stream. The default is **`40`**.

Tag	Type	Description
`pat_interval`	integer	The number of milliseco nds between the program association tables (PAT) in the output manifest. The default is **`100`**.
`pmt_interval`	interval	The number of milliseconds between the program map tables (PMT) in the output manifest. The default is **`100`**.

The `stream_set` Element

The `stream_set` element holds information about how Delta handles the audio, video, and subtitles in a stream. The following table describes the settings for the `stream_set` element.

Tag	Type	Description
`stream_index`	integer	An optional number entered in each stream set in the filter to sort the sets in relation to each other. A lower number puts the set higher in the list. Numbers can be skipped. For example, if you later remove a stream set from the filter, there is no need to renumber the other sets.
`audio_tracks`	object	This tag contains informati on about the audio tracks to include in the stream set. To specify more than one

Tag	Type	Description
		audio track in the stream set, include multiple `audio_track` objects. See the table below for the tags to enter in the `audio_track` .
`video_tracks`	object	This tag contains information about the video tracks to include in the stream set. To specify more than one video track in the stream set, include multiple `video_track` objects. See the table below for the tags to enter in the `video_track` .
`subtitle_tracks`	object	This tag contains information about the subtitle or captions tracks to include in the stream set. To specify more than one subtitle or captions track in the stream set, include multiple `subtitle_track` objects. See the table below for the tags to enter in the `subtitle_track` .

The `video_track`, `audio_track`, and `subtitle_track` Elements

The following table describes the settings for the `video_track`, `audio_track`, and `subtitle_track` elements.

Tag	Type	Description
`track_id`	integer	Read-only.

Tag	Type	Description
		The Delta-assigned numeric value for the track.
`track_index`	integer	A number used to sort the tracks in relation to each other if the stream set contains more than one track of the same type (video, audio, or subtitle). Enter a number in each *xx* track. A lower number puts the track higher in the list. Numbers can be skipped. For example, if you later remove a track from the set, there is no need to renumber the other tracks.

PUT: Create an MSS Package Output Filter

To create a Microsoft Smooth Streaming (MSS) package output filter on a content object, send a PUT request. The following sections describe how to format the request and provide examples of the request and response.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID
```

Body of HTTP

The body of your request is XML content consisting of one `content` element that holds:

- One `filters` container and one `filter` element that holds several tags, and also holds:
  - One `filter_settings` element that holds several tags.

- One `stream_sets` container and zero or one `stream_set` element (zero only if `use_default_stream_sets` is **`true`**). The `stream_set` element is required for DVB-Text and DVB-Sub subtitles.

  If present, the stream_set element holds several tags and also holds:

  - One `video_tracks` container and zero or more `video_track` elements that hold several tags.
  - One `audio_tracks` container and zero or one `audio_track` element that holds several tags.
  - One `subtitle_tracks` container and zero or one `subtitle_track` element that holds several tags.

Request Example

This request modifies the existing content entity that has ID 39 to add an MSS output filter. The request doesn't include a `parent_id` because it is a top-level filter. It includes an `output_url` (to create a custom endpoint that contains `movie21`). It includes one stream set that holds one audio track and one video track.

```
PUT http://10.24.34.2:8080/contents/39
---------------------------------------
<content>
  <filters>
    <filter>
      <endpoint>true</endpoint>
      <output_url>movie21</output_url>
      <filter_type>mss_package</filter_type>
      <filter_settings>
        <fragment_duration>2</fragment_duration>
        <index_duration>60</index_duration>
      </filter_settings>
      <stream_sets>
        <stream_set>
          <stream_index>10</stream_index>
          <video_tracks>
            <video_track>
              <track_index>0</track_index>
              <track_id>1</track_id>
            </video_track>
          </video_tracks>
          <audio_tracks>
```

```
            <audio_track>
              <track_index>1</track_index>
              <track_id>2</track_id>
            </audio_track>
          </audio_tracks>
        </stream_set>
      </stream_sets>
    </filter>
  </filters>
</content>
```

Response

The response repeats back the filter that you created, with the addition of the following tags:

- `id`: Unique IDs for the filter and each sub-element as described in this section.
- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI, as follows:

  ```
  http://<Delta IP address>:8080/out/i/<filter ID>..ism/Manifest
  ```

  where `filter ID` is the ID for this filter, assigned once the filter is created.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if `output_uri` contains a value. Contains the custom endpoint URI, as follows:

  ```
  http://<Delta IP address:8080>/out/u/<output_url>..ism/Manifest
  ```

This example response is a representation. The `default_endpoint` now shows a value. The `filter_settings` shows the default values. The `filter`, `filter_settings`, `stream_set`, `video_track`, and `audio_track` elements are all assigned unique IDs. Your results may vary.

```
<?xml version="1.0" encoding="UTF-8"?>
<contents href="/contents" product="Delta" version="2.3.0.123456">
  <content href="/contents/39" product="Delta" version="2.3.0.123456">
    <filters href="/contents/39/filters" product="Delta" version="2.3.0.123456">
      <filter href="/contents/39/filters/29" product="Delta" version="2.3.0.123456">
        <id>29</id>
        <parent_id/>
        <endpoint>true</endpoint>
        <default_endpoint_uri>http://10.24.34.2/out/i/29.ism/Manifest</
default_endpoint_uri>
```

```
        <custom_endpoint_uri>http://10.24.34.2/out/u/movie21.ism/Manifest</
custom_endpoint_uri>
        <output_urlmovie21</output_url>
        <filter_type>mss_package</filter_type>
        <filter_settings>
          <id>1</id>
          <fragment_duration>2</fragment_duration>
          <index_duration>60</index_duration>
        </filter_settings>
        <stream_sets>
          <stream_set>
            <id>27</id>
            <stream_index>0</stream_index>
            <iframe_only>false</iframe_only>
            <video_tracks>
              <video_track>
                <id>43</id>
                <track_index>0</track_index>
                <track_id>54</track_id>
              </video_track>
            </video_tracks>
            <audio_tracks>
              <audio_track>
                <id>46</id>
                <track_index>3</track_index>
                <track_id>56</track_id>
              </audio_track>
            </audio_tracks>
            <subtitle_tracks/>
          </stream_set>
        </stream_sets>
      </filter>
    </filters>
  </content>
</contents>
```

Topics

- The filter Element
- The filter_settings Element
- The stream_set Element

The `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only.

Tag	Type	Description
		The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. Delta automatically creates this customized address, including the `output_url` . Delta stores the address in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.

Tag	Type	Description
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
`fragment_duration`	integer	The length of fragments (in seconds) that Delta generates. Delta rounds actual fragments to the

Tag	Type	Description
		nearest multiple of the source content fragment length. The default is the value from the ingested content.
`index_duration`	integer	The length of the live manifest (in seconds). Delta ignores this parameter for VOD content.
`fragment_lookahead`	integer	The number of fragments that Delta must keep in the buffer. Applies to live content only. The default value is **2**. When you specify a different value, it must be greater than **0**.
`enable_events`	integer	This tag specifies how Delta handles `events` tags in the output manifest. • **`true`** is the default. The `events` tag is embedded in the URL of the output manifest. • **`false`** means that the `events` tag is embedded in the output manifest.

The `stream_set` Element

The `stream_set` element holds information about how Delta handles the audio, video, and subtitles in a stream. The following table describes the settings for the `stream_set` element.

Tag	Type	Description
`stream_index`	integer	An optional number entered in each stream set in the filter to sort the sets in relation to each other. A lower number puts the set higher in the list. Numbers can be skipped. For example, if you later remove a stream set from the filter, there is no need to renumber the other sets.
`audio_tracks`	object	This tag contains informati on about the audio tracks to include in the stream set. To specify more than one audio track in the stream set, include multiple `audio_tra ck` objects. See the table below for the tags to enter in the `audio_track` .
`video_tracks`	object	This tag contains informati on about the video tracks to include in the stream set. To specify more than one video track in the stream set, include multiple `video_tra ck` objects. See the table below for the tags to enter in the `video_track` .
`subtitle_tracks`	object	This tag contains information about the subtitle or captions

Tag	Type	Description
		tracks to include in the stream set. To specify more than one subtitle or captions track in the stream set, include multiple `subtitle_track` objects. See the table below for the tags to enter in the `subtitle_track` .

The `video_track`, `audio_track`, and `subtitle_track` Elements

The following table describes the settings for the `video_track`, `audio_track`, and `subtitle_track` elements.

Tag	Type	Description
`track_id`	integer	Read-only. The Delta-assigned numeric value for the track.
`track_index`	integer	A number used to sort the tracks in relation to each other if the stream set contains more than one track of the same type (video, audio, or subtitle). Enter a number in each *xx* track. A lower number puts the track higher in the list. Numbers can be skipped. For example, if you later remove a track from the set, there is no

Tag	Type	Description
		need to renumber the other tracks.

PUT: Create a Passthrough Package Output Filter

To create a Passthrough package output filter on a content object, send a PUT request. This filter serves the content as it was formatted by the upstream system. The following sections describe how to format the request and provide examples of the request and response.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID
```

Body of HTTP

The body of your request is XML content consisting of one `content` element that holds:

- One `filters` container and one `container` element that holds several tags.

Request Example

This request modifies the existing content entity that has ID 32 to add a Passthrough output filter. The request doesn't include a `parent_id` because it is a top-level filter. It includes an `output_url` (to create a custom endpoint that contains `060915`).

```
PUT http://10.24.34.2:8080/contents/32
---------------------------------------
<content>
  <filters>
    <filter>
      <endpoint>true</endpoint>
      <output_url>060915</output_url>
      <filter_type>passthrough</filter_type>
    </filter>
  </filters>
</content>
```

Response

The response repeats back the filter you created, with the addition of the following tags:

- `id`: Unique IDs for the filter and each sub-elements as described in this section.
- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI as follows:

  ```
  http://Delta IP address:8080/out/i/filter ID/filter name.extension
  ```

  where

  - `filter ID` is the ID for this filter, assigned once the filter is created.
  - `file name` is the filename for the ingested content.
  - `extension` is the extension of the ingested content, either `m3u8` or `mp4`.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if `output_uri` contains a value. Contains the custom endpoint URI, as follows:

  ```
  http://Delta IP address/out/u/output_url.extension
  ```

This example response is a representation. The `default_endpoint` now shows a value. The `filter_settings` shows the default values. The `filter`, `filter_settings`, `stream_set`, `video_track` and `audio_track` elements are all assigned unique IDs. Your results may vary.

```
<?xml version="1.0" encoding="UTF-8"?>
<contents href="/contents" product="Delta" version="2.3.0.123456"
  <content href="/contents/34" product="Delta" version="2.3.0.123456"
    <filters href="/contents/34/filters" product="Delta" version="2.3.0.123456"
      <filter href="/contents/34/filters/32" product="Delta" version="2.3.0.123456">
      <id>32</id>
      <parent_id/>
      <endpoint>true</endpoint>
      <default_endpoint_uri>http://10.24.34.2/out/i/32/comedy_2.m3u8</
default_endpoint_uri>
      <custom_endpoint_uri>http://10.24.34.2/out/p/32/060915.m3u8</
custom_endpoint_uri>
      <output_url>060915</output_url>
      <filter_type>passthrough</filter_type>
    </filter>
  </filters>
</content>
```

Topics

- The filter Element

The `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only.

Tag	Type	Description
		The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. Delta automatically creates this customized address, including the `output_url` . Delta stores the address in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.

Tag	Type	Description
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

Working with Tracks: GET Tracks

When Delta ingests an asset, it automatically detects the video, audio, and subtitle tracks. Each track is set up with a unique ID.

You can get a list of all the tracks for a specified content object in Delta. You can then include the ID for a specific track in the `stream_set` element for a package output filter that you are creating. See the sections on the `stream_set` element in each package output filter sections, as described in Creating Package Output Filters in AWS Elemental Delta.

HTTP URL

```
GET http://Delta IP address:8080/contents/39/tracks
```

Response

The body of your request is XML content consisting of one `content` element that holds:

- An HREF that specifies the unique ID of the content and the product and version installed on the node.
- XML content consisting of one `tracks` element that contains:
  - Zero or more `track` elements, each with several tags. The tags are slightly different for video, audio, and subtitle tracks. See the sections below.

Topics

- The track Element for Audio
- The track Element for Subtitles
- The track Element for Video

The `track` Element for Audio

The following tags are in the `track` element for audio.

Tag	Type	Description
`id`	integer	The unique ID for this track, assigned by Delta.
`bits_per_sample`	integer	
`channel_count`	integer	
`codec`	string	The codec of the track. Either **`AACL`**, **`AC3`**, or **`EAC3`**.
`language`	string	The language of the track.
`sample_rate`	integer	The numerator for the sample rate of this track. For example, **`44100`** means 44100 samples per second.

Tag	Type	Description
`sample_rate_denominator`	integer	The denominator for the sample rate of this track. This value indicates the timeframe for the number of samples (`sample_rate`). Typically this is **1**, which means x number of samples per 1 second.
`track_type`	string	The type of track. In this case, **`audio`**.

The `track` Element for Subtitles

The following tags are in the `track` element for subtitles and captions.

Tag	Type	Description
`id`	integer	The unique ID for this track, assigned by Delta.
`codec`	string	The codec of the track. Either **`DFXP`**, **`TTML`**, or **`WebVTT`**.
`language`	string	The language of the track.
`pid`	integer	The PID of this track.
`stream_index`	integer	The stream index for this track.
`track_type`	string	The type of track. In this case, **`subtitle`**.

The `track` Element for Video

The following tags are in the `track` element for video.

Tag	Type	Description
`id`	integer	The unique ID for this track, assigned by Delta.
`bitrate`	integer	The bitrate of the track.
`codec`	string	The codec of the track. Either **H264** or **H265**.
`framerate_denominator`	integer	The denominator for the framerate. For example, 30 frames per second is a numerator of 30 and a denominator of 1. Similarly, 20.334 frames per second is a numerator of 20334 and a denominator of 1000. Default is 1.
`framerate_numerator`	integer	See `framerate_denominator` . Default is 30.
`height`	integer	The resolution height of the video track.
`pid`	integer	The PID of this track.
`stream_index`	integer	The stream index for this track.

Tag	Type	Description
`track_type`	string	The type of track. In this case, **`video`**.
`width`	integer	The resolution width of the video track.

Creating Access Restriction Filters in AWS Elemental Delta

Access restriction output filters are those that control the downstream player's ability to access the output content. The filters are described in the following sections.

Topics

- PUT: Create a DRM Output Filter
- PUT: Create an Authentication Output Filter
- PUT: Create a Whitelist or Blacklist Output Filter
- PUT: Create a User Agent Output Filter
- Working with User Agent Presets

PUT: Create a DRM Output Filter

Digital rights management (DRM) output filters contain encryption information so that access to the content can be controlled by a DRM solution. To create the output filter, send a PUT request for the content entity that corresponds to the content that DRM applies to. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID
```

Body of HTTP

The body of your request is XML content. The structure of the XML content depends on the type of filter (`filter_type` tag in each filter element in the body).

See the following sections for information and request examples for each DRM output filter.

- Common Encryption (CENC) Output Filter
- Flash Access Output Filter
- HLS Encryption Output Filter

- PlayReady Output Filter

Response

The response repeats back all the possible tags in the filter you created (including those you did not set) with the addition of:

- `id`: Unique IDs for the filter and each sub-element as described in this section.
- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI as follows:

```
http://Delta IP address:8080/out/i/filter ID.extension
```

  where:

  - `filter ID` is the ID for this filter, assigned once the filter is created.
  - `extension` is different for each output filter type.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if the `output_uri` tag contains a value. The `custom_endpoint_uri` contains the custom endpoint URI as follows:

```
http://Delta IP address:8080/out/u/output_url.m3u8
```

The response is identical to the response to a Viewing Filters List: GET Output Filters List.

Common Encryption (CENC) Output Filter

The following sections describe how to format the request to add a Common Encryption (CENC) output filter to a contents object.

Body of HTTP

The body of your request is XML content consisting of one `filters` container that holds one `filter` element. The `filter` element holds the following:

- One `filter_settings` element that holds the:
  - `filter ID`
  - `keyprovider_type`
  - `keyprovider_settings` element that captures information about the keyprovider that you use.

Request Example

This request modifies the existing content entity that has ID 79 to add a Common Encryption output filter with Widevine as the key provider type.

```
PUT http://10.24.34.2:8080/contents/79
---------------------------------------
<content>
    <filters>
        <filter>
            <parent_id>47</parent_id>
            <filter_type>common_encryption</filter_type>
            <endpoint>false</endpoint>
            <output_url nil="true"/>
            <description nil="true"/>
            <filter_settings>
                <keyprovider_type>irdeto</keyprovider_type>
                <keyprovider_settings>
                    <content_id>content45</content_id>
                    <account_id>568974523</account_id>
                    <sub_content_type nil="true"/>
                    <use_https>false</use_https>
                    <content_key>generate_new_key</content_key>
                    <key_source>widevine_key_server</key_source>
                    <key_id nil="true"/>
                    <key_seed nil="true"/>
                    <domain_service_id nil="true"/>
                    <request_cenc_key>true</request_cenc_key>
                    <request_playready_key>false</request_playready_key>
                    <service_url>
                        <uri>http://url.com</uri>
                        <username>user456</username>
                        <password>password8</password>
                    </service_url>
                    <la_url nil="true"/>
                </keyprovider_settings>
            </filter_settings>
        </filter>
    </filters>
</content>
```

Topics

- The CENC filter Element
- The CENC filter_settings Element
- The CENC keyprovider_settings Element

The CENC `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.

Tag	Type	Description
`default_endpoint_uri`	string	Read-only. The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. This customized address including the `output_url` is automatically create and stored in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.

Tag	Type	Description
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).

Tag	Type	Description
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The CENC `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
`content_id`	string	This setting provides the keyprovider server with an ID to identify the Delta content. Each `content_id` value you assign in Delta should be unique for this keyprovider server.
`key_rotation_count`	integer	The AES-128 encryption key rotates after this many segments. Set to **`0`** (default) to use the same key throughout the entire encoding session (static keys).
`keyprovider_type`	string	This setting is the keyprovider for CENC. See The CENC `keyprovider_settings` Element for keyprovider-specific tags: • `widevine` • `generic_cenc` • `nagra` • `irdeto` • `speke`

Tag	Type	Description
`pssh_version`	integer	This is the specification version that Delta uses for the protection system-specific header (PSSH). The version determines how the PSSH data is formatted coming out of Delta. Valid values are **0** (default) or **1**.

The CENC `keyprovider_settings` Element

The following sections describe the settings for each keyprovider.

Generic Keyprovider

The following table shows the tags to use in the `keyprovider_settings` object when using a generic keyprovider.

Tag	Type	Description
`key_value`	hexadecimaladecimal string	The 16-byte hexadecim aladecimal value of the encryption key.
`kid`	hexadecimaladecimal string	The 16-byte hexadecim aladecimal value of the key ID.
`pssh`	string	PSSH is a base 64-encoded string.
`protection_uuid`	string	The unique identifier for the content protection system.

Irdeto

The following table shows the tags to use in the `keyprovider_settings` object when using Irdeto for your keyprovider.

Tag	Type	Description
`service_url`	server	Specifies the location of the Irdeto server. Both a URL and login credentials are required. For contents of the server type, see Server Type Elements.
`account_id`	string	Your account on the Irdeto Control server.
`content_id`	string	This tag identifies the Delta content in Irdeto Control. Each `content_id` value you assign in Delta should be unique for this Irdeto Control.
`username`	string	The username used to authenticate the requesting device to the keyprovider.
`password`	string	The password to authenticate the requesting device to the keyprovider.
`request_CENC_key`	boolean	**`true`** to receive CENC PSSH data from the key server. Must be **`true`** if `request_playready_key` is **`false`**.

Tag	Type	Description
`request_playready_key`	boolean	**`true`** to receive Playready PSSH data from the key server. Must be **`true`** if `request_CENC_key` is **`false`**.

Nagra

Use the following tags in the `keyprovider_settings` object when using Nagra for your keyprovider.

Tag	Type	Description
`content_id`	string	Identifies the Delta content on the Nagra server. Each `content_id` value you assign in Delta should be unique for this Nagra server.
`server_url`	string	Location of the Nagra server.

Secure Package and Encoder Key Exchange (SPEKE)

Use the following tags in the `keyprovider_settings` object when using Secure Package and Encoder Key Exchange (SPEKE) for your keyprovider.

Tag	Type	Description
`system_ids`	string	Unique identifier for your DRM solution provider. For a list of common system IDs, see https://dashif.org/identifiers/content_protection/. If

Tag	Type	Description
		you don't know your ID, ask your DRM solution provider.
`content_id`	string	Identifies the Delta content on the Nagra server. Each `content_id` value you assign in Delta should be unique for this Widevine server.
`speke_server`	object	Holds information specific to the Secure Package and Encoder Key Exchange (SPEKE) server. See the following table.

`speke_server` Settings

Include the following required settings in the `speke_server` object.

Tag	Type	Description
`uri`	string	The location of the SPEKE server that provides your DRM keys.
`username`	string	The username when credentia ls are required to access a file or publishing point.
`password`	string	The password when credentia ls are required to access a file or publishing point.

Widevine

Use the following tags in the `keyprovider_settings` object when using Widevine for your keyprovider.

Tag	Type	Description
`license_url`	string	The URL for the keyprovider server.
`content_id`	string	The identifier for the Delta content on the Nagra server. Each `content_id` value you assign in Delta should be unique for this Widevine server.
`provider_id`	string	The “signer” for the keyprovid er requests.
`provider_key`	hexadecimaladecimal string	The AES key for signing keyprovider requests. This is a 256-bit hexadecimaladecima l value represented by a 64-character string.
`provider_iv`	hexadecimaladecimal string	The AES Initialization Vector (IV) for signing keyprovider requests. This is a 128-bit hexadecimaladecimal value represented by a 32-character string.
`key_rotation_count`	integer	The number of segments before the system requests a new key from the keyprovid er. Set to **`0`** to use the same

Tag	Type	Description
		key for the duration of the content.
`reuse_last_key`	boolean	• **`true`**: if the key provider becomes unreachable, the stream is encrypted using the last key obtained from the key provider. • **`false`**: if the key provider becomes unreachable, the request to encrypt fails and the content is not available to client players.
`request_widevine_key`	boolean	• **`true`**: if you want a Widevine key from the Widevine keyprovider. • **`false`**: if you do not want a Widevine key. You can enter **`true`** in both `request_playready_key` and `request_widevine_key` or enter **`true`** in one tag only. You cannot enter **`false`** in both tags.

Tag	Type	Description
`request_playready_key`	boolean	• **`true`**: if you want a Playready key from the Widevine keyprovider. The keyprovider must support Playready keys. • **`false`**: if you do not want a Playready key. You can enter **`true`** in both `request_playready_key` and `request_widevine_key` or enter **`true`** in one tag only. You cannot enter **`false`** in both tags.

Flash Access Output Filter

The following sections describe how to format the request to add a Flash Access output filter to a contents object.

Body of HTTP

The body of your request is XML content consisting of one `filters` container that holds one `filter` element. The `filter` element contains the following:

- One `filter_settings` element that holds several tags.

Request Example

This request modifies the existing content entity that has ID 39 to add a Flash Access output filter. In this example, for most of the tags that specify a path to a file, the path is to a remote share that you have mounted to the Delta node: therefore, the path is `/data/server/xx`.

```
PUT http://10.24.34.2:8080/contents/39
---------------------------------------
<content>
```

```
    <filters>
        <filter>
            <parent_id>81</parent_id>
            <endpoint>true</endpoint>
            <filter_type>flash_access</filter_type>
            <use_default_stream_sets>true</use_default_stream_sets>
            <filter_settings>
                <encrypt_audio>true</encrypt_audio>
                <encrypt_data>true</encrypt_data>
                <encrypt_video>true</encrypt_video>
                <video_encrypt_level>3</video_encrypt_level>
                <policy_file>/data/server/flash_access/policy_file.pol</policy_file>
                <swf_identifiers>/data/server/MyWhitelistedPlayers.txt</
swf_identifiers>
                <use_phds>false</use_phds>
                <content_encryption_key>/data/server/flash_access/
content_encryption_key.key</content_encryption_key>
                <content_id>MOV-347465</content_id>
                <generate_cek>true</generate_cek>
                <key_server_certificate>/data/server/flash_access/elemental-
keyServerCertificate.der</key_server_certificate>
                <license_server_certificate>/data/server/flash_access/elemental-
license.der</license_server_certificate>
                <license_server_credential>/data/server/flash_access/
licenseServerCredentials.txt</license_server_credential>
                <license_server_credential_password>secret</
license_server_credential_password>
                <license_server>http://example.com:8080/flashaccessserver/
elemental</license_server>
                <packager_credential>/data/server/flash_access/elemental-
license.pfx</packager_credential>
                <packager_credential_password>secret</packager_credential_password>
                <recipient_certificate>/data/server/flash_access/
recipientCertificate.der</recipient_certificate>
                <transport_certificate>/data/server/flash_access/
transportCertifiate.der</transport_certificate>
            </filter_settings>
        </filter>
    </filters>
</content>
```

Topics

- The Flash Access filter Element
- The Flash Access filter_settings Element

The Flash Access `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only.

Tag	Type	Description
		The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. This customized address including the `output_url` is automatically create and stored in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.

Tag	Type	Description
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The Flash Access `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
`use_phds`	boolean	• **`true`** is the default and means that Protected HDS or Protected HLS are enabled on applicable content.

Tag	Type	Description
		• **`false`** means that Delta applies full Flash Access encryption using the server you specify in `license_server`. Additional tags are required. See the following sections.
`encrypt_audio`	boolean	**`true`** is the default and means that audio data is encrypted. At least one of `encrypt_audio`, `encrypt_video`, or `encrypt_data` must be enabled.
`encrypt_data`	boolean	**`true`** is the default and means that stream metadata is encrypted.
`encrypt_video`	boolean	**`true`** is the default and means that video data is encrypted.

Tag	Type	Description
`video_encrypt_level`	string	A tag that specifies the degree of partial encryption to apply. • **`low`** is the default and means that the lowest amount of partial encryptio n should be applied. A subset of the samples (like video keyframes) is encrypted. • **`medium`** means that a medium amount of partial encryption should be applied. • **`high`** means that full encryption should be applied.
`policy_file`	string	Path to the file that contains the policy file.
`swf_identifiers`	string	Path to a file of hashes of SWF players that you have approved as valid players for this content. Use the Adobe Media Server Whitelist tool to generate this file.

Full Flash Access Settings

The following tags apply only when `use_phds` is **`false`**.

Tag	Type	Description
`generate_cek`	boolean	This tag controls how the content encryption key (CEK) is generated. • **`true`** is the default and means that the `content_encrytption_key` and `content_id` are combined to generate a unique Content Encryption Key (CEK). The `content_encryption_key` can be a file of arbitrary length. • **`false`** means that the `content_encryption_key` is used directly as the CEK. This key must be 16 bytes (128 bits) long.
`content_id`	string	Complete only if a common key is in use. The `content_id` is used with the `content_encryption_key` tag to generate a CEK.
`packager_credential`	string	The path to the credentials for the Adobe packager.
`packager_credential_password`	string	The password for the credential file identified in the `packager_credential` tag.
`transport_certificate`	string	The transport certificate, in DER format.

Tag	Type	Description
`license_server`	string	The URL of the Adobe Access license server used for protecting content.
`key_server_certificate`	string	The path to the certificate required to support an embedded (non-chained) license with Remote Key Delivery. Must be accessible to the Apache user.
`license_server_certificate`	string	The path to the unique certificate file obtained from Adobe that identifies the license server, in DER format.
`recipient_certificate`	string	The path to the certificate that uniquely identifies the recipient machine and client instance.
`license_server_credential`	string	The path and name for the credential file for the Adobe Access license server.
`license_server_credential_password`	string	The password for the credential file for the Adobe Access license server.
`content_encryption_key`	string	Path and filename for the cryptographic key used to encrypt the content.

HLS Encryption Output Filter

The following sections describe how to format the request to add an HLS Encryption output filter to a contents object.

Body of HTTP

The body of your request is XML content consisting of one `filters` container that holds one `filter` element. The `filter` element contains the following:

- One `filter_settings` element that holds several tags, including the:
  - `filter ID`
  - `keyprovider_type`
  - `keyprovider_settings` element that captures information about the keyprovider that you use. This element does not exist if the keyprovider is VOSP. The VOSP keyprovider tags exist inside the `filter_settings` element rather than within the `keyprovider_settings`.

Request Example

This request modifies the existing content entity that has ID 39 to add an HLS Encryption output filter with Irdeto as the key provider type (specified about halfway down the body). In this example, `static_key` is set to **`true`**. See the Playready example (below) for sample values when `static_key` is **`false`**.

```
PUT http://10.24.34.2:8080/contents/39
---------------------------------------
<content>
    <filters>
        <filter>
            <parent_id>66</parent_id>
            <endpoint>true</endpoint>
            <filter_type>hls_encryption</filter_type>
            <use_default_stream_sets>true</use_default_stream_sets>
            <filter_settings>
                <constant_iv>37543967320876342873939654304912 8</constant_iv>
                <encryption_type>AES-128</encryption_type>
                <repeat_ext_x_key>true</repeat_ext_x_key>
                <iv_follows_segment_number>false</iv_follows_segment_number>
                <key_format_versions>1</key_format_versions>
                <key_format>identity</key_format>
```

```
                <keyprovider_type>irdeto</keyprovider_type>
                <keyprovider_settings>
                    <key_source>key_server</key_source>
                    <id>1</id>
                    <static_key>true</static_key>
                    <account_id>myIrdetoAccount</account_id>
                    <content_id>MOV-13094</content_id>
                    <content_key>generate_new_key</content_key>
                    <domain_service_id>eird93uu-3gyj83zis-opp9-jdgeti387909</
domain_service_id>
                    <key_id>eird93uu-3gyj83zis-opp9-jdgeti387909</key_id>
                    <key_seed>39eHgu790RtJdkgurEOeju0</key_seed>
                    <service_url>
                        <password>secret</password>
                        <uri>https://example.com/livedrmservice/
livedrmservice.asmx</uri>
                        <username>myUserName</username>
                    </service_url>
                    <sub_content_>SSPlayReady</sub_content_>
                    <use_https>false</use_https>
                    <use_rotating_keys>false</use_rotating_keys>
                </keyprovider_settings>
            </filter_settings>
        </filter>
    </filters>
</content>
```

Topics

- The HLS Encryption filter Element
- The HLS Encryption filter_settings Element
- The HLS Encryption keyprovider_settings Element

The HLS Encryption `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only. The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. This customized address

Tag	Type	Description
		including the `output_url` is automatically create and stored in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.

Tag	Type	Description
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The HLS Encryption `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

With the `filter_settings` for the HLS Encryption filter, a given tag may go in one of two places, depending on the keyprovider being used: for some keyproviders, the tag goes in the `filter_settings` element, while, for other keyproviders, it goes in the `keyprovider_settings` sub-element of `filter_settings`.

Tag	Type	Description
`encryption_type`	string	The type of encryption, either **`AES-128`** or **`Sample-AES`** . **`Sample-AES`** is valid only if the `keyprovider_settings` specifies **`1mainstream`** , **`castlabs`**, **`irdeto`**, **`generic_keyprovider`** , or **`static_hls`** .
`key_format`	string	The key format value, either: • **`identity`** (default), or • A reverse DNS string. For example, com.example.sample.
`key_format_versions`	string	The version of the `key_format` that the keyserver supports. Can be either: • A single positive integer version value (for example, **`1`**), or • A slash-delimited list of version values (for example, **`1/2/3`**)
`key_rotation_count`	integer	The AES-128 encryption key rotates after this many segments. Set to **`0`** to use the same key throughout the entire encoding session. This

Tag	Type	Description
		parameter is ignored when the `keyprovider_type` is **`conax`**, **`irdeto`**, or **`1mainstream`** .
`repeat_ext_x_key`	boolean	When **`true`**, repeats the EXT-X-KEY directive for every media segment. This may increase client requests to the DRM server.
`iv_follows_segment_number`	boolean	The IV (Initialization Vector) is a 128-bit number used with the key for encrypting blocks. • **`true`** is the default and causes the IV to change every segment (to match the segment number). • **`false`** means the value in `constant_iv` is used.
`constant_iv`	hexadecimaladecimal string	A 128-bit, 16-byte hexadecim aladecimal value represented by a 32-character string, used as the IV for encryption. Required when `iv_follows_segment_number` is **`false`**.

Tag	Type	Description
`keyprovider_type`	string	The type of keyprovider. Available options include: • **`castlabs`** • **`conax`** • **`discretix`** • **`generic_keyprovider`** • **`irdeto`** • **`one_mainstream`** • **`nagra`** • **`piksel`** • **`secure_media`** • **`self_generated`** • **`speke`** • **`static_hls`** • **`verimatrix`** • **`vosp`**
`keyprovider_settings`	object	The object that holds the settings specific to your keyprovider. See The HLS Encryption `keyprovider_settings` Element for settings by keyprovider. Note that VOSP information is captured in the `filter_settings` object.

The HLS Encryption `keyprovider_settings` Element

The following sections describe the settings for each keyprovider.

castLabs

The following table describes the tags that are specific for the castLabs server.

Tag	Type	Description
`content_id`	string	The tag that provides the CastLabs server with an ID to identify the Delta content. Delta generates the encryption key for this content ID and provides it in the key request. To avoid key/content collisions, each `content_id` value you assign in Delta should be unique for this CastLabs server. If it is not unique, then a validation error is generated and the filter is not created. If your workflow requires duplicate `content_id` tags, then use the key seed option.
`key_seed`	string	The tag that contains the base 64 key seed that you retrieved from the CastLabs administrative interface. When a key seed is provided, Delta does not generate an encryption key and instead includes the key seed in the key request. The key seed allows for unavoidable duplicate content IDs, such as with output templates.

Tag	Type	Description
		Required on VOD output templates.
`server`	location	The tag that specifies the location of the Castlabs server. For contents of the location type, see Location Type Elements.
`username`	string	The username used to authenticate to the keyprovid er.
`password`	string	The password used to authenticate to the keyprovid er.
`merchant_id`	string	A merchant ID provided by DRMtoday.

Conax

The following table describes the tags that are specific for the Conax server.

Tag	Type	Description
`content_id`	string	The tag that provides the Conax server with an ID to identify the Delta content. Each `content_id` value you assign in Delta should be unique for this Conax server.
`server`	location	The tag that specifies the location of the Conax server.

Tag	Type	Description
		A URL and login credentials are required. For contents of the location type, see Location Type Elements.
`username`	string	The username used to authenticate to the keyprovider.
`password`	string	The password used to authenticate to the keyprovider.
`resource_type`	string	The Conax resource type. Use: • **`vod`** for VOD content or after Live to VOD filters, or • **`dtv`** for output live content.

Discretix

The following table describes the tags that are specific for the Discretix server.

Tag	Type	Description
`static_key`	boolean	• **`false`** is the default and means that Delta fetches the `key_id`, `key_seed`, and `service_id` from the Discretix server. • **`true`** means that you provide these values.

Tag	Type	Description
		See the following tables for additional required tags.

Static Key Source Settings

Use the following tags when `static_key` is **`true`**.

Tag	Type	Description
`key_id`	string	The tag that specifies a key ID. Must be a valid GUID.
`key_seed`	string	The tag that contains a base 64-encoded key seed.
`service_id`	string	The VOSP Service ID.
`license_url`	string	The URL for the license acquisition web service.

Key Server Source Settings

Use the following tags when `static_key` is **`false`**.

Tag	Type	Description
`content_id`	string	The tag which provides the Conax server with an ID to identify the Delta content. Each `content_id` value you assign in Delta should be unique for this Conax server.
`license_url`	string	The URL for the license acquisition web service.

Tag	Type	Description
`server`	server	The tag that specifies the location of the Discretix server. A URL and login credentials are required. For contents of the location type, see Location Type Elements.

Generic Keyprovider

The following table describes the tags that are specific for a generic keyprovider.

Tag	Type	Description
`resourceid`	string	The tag used by the Generic Keyprovider to identify the content.
`server`	location	The tag that specifies the location of the Generic Keyprovider server. A valid URI is required. Optional username and password are used if the keyprovider requires authentication. For contents of the location type, see Location Type Elements.
`reuse_last_key`	boolean	• **`false`** is the default and means that, if the media server becomes unreachab le, the endpoint on this DRM filter and endpoints

Tag	Type	Description
		downstream of this DRM filter are not accessible. • **`true`** means that the output is encrypted using the last key obtained from the server in the event that it becomes unreachable.
`resource_type`	string	• **`vod`** for output VOD content or after Live to VOD filters, or • **`dtv`** for output live content

Irdeto

The following table describes the tags that are specific for the Irdeto server.

Tag	Type	Description
`service_url`	server	This tag applies when the `encryption_type` is **`Sample-AES`** only. A tag that specifies the location of the Irdeto server. Both a URL and login credentials are required. For contents of the server type, see Server Type Elements.
`account_id`	string	A tag that applies when the `encryption_type` is **`Sample-AES`** only.

Tag	Type	Description
		Your account on the Irdeto Control server.
`content_id`	string	A tag that applies when the `encryption_type` is **`Sample-AES`** only. The identifier for the Delta content in Irdeto Control. Each `content_id` value you assign in Delta should be unique for this Irdeto Control.
`username`	string	A tag that applies when the `encryption_type` is **`Sample-AES`** only. The user name used to authenticate to the keyprovid er.
`password`	string	A tag that applies when the `encryption_type` is **`Sample-AES`** only. The password used to authenticate to the keyprovid er.

Tag	Type	Description
`key_source`	string	A tag that specifies how the key is being provided. Available options: • `static`: you provide `key_id`, `key_seed`, `la_url`, and `domain_service_id` . • `key_server` : values are fetched from the Irdeto server. • `ske_key_server` : values are fetched from the Irdeto SKE server. • `ca_key_server` : values are fetched from the Irdeto server with a CA protection_type. See the following sections for required tags based on the `key_source` .

Static Key Source Settings

Use the following tags in an HLS Encryption (AES-128) filter when `key_source` is **`static`**.

Tag	Type	Description
`key_id`	string	This tag specifies a key ID. Must be a valid GUID.

Tag	Type	Description
`key_seed`	string	This tag contains a base 64-encoded key seed.
`domain_service_id`	string	This is your Service ID. Must be a valid GUID.
`account_id`	string	This tag is your account on the Irdeto Control server.
`content_id`	string	This tag identifies the Delta content in Irdeto Control. Each `content_id` value you assign in Delta should be unique for this Irdeto Control.
`sub_content_type`	string	This tag specifies the sub-content type to be associated with the output group. Either: • `default` • `SSPlayReady` • `HLSPlayReady` , or • Other customer-supported value.
`la_url`	string	This tag is the address used to access license acquisition.

Key Server Source Settings

Use the following tags in an HLS Encryption (AES-128) filter when `key_source` is **`key_server`**.

Tag	Type	Description
`service_url`	location	This tag specifies the location of the Irdeto server. Both a

Tag	Type	Description
		URL and login credentials are required. For contents of the location type, see Location Type Elements.
`account_id`	string	This tag is your account on the Irdeto Control server.
`content_id`	string	This tag identifies the Delta content in Irdeto Control. Each `content_id` value you assign in Delta should be unique for this Irdeto Control.
`content_key`	string	This tag specifies when a new key should be generated: • `generate_new_key` : generate a key at the start of encoding. • `use_last_key` : the last key used in the encoding session.
`use_https`	boolean	• **`false`** is the default and means that Delta uses basic HTTP when communicating with the License Acquisition URL (`la_url` tag). • **`true`** means that Delta uses HTTPS in making requests to the License Acquisition URL.

Tag	Type	Description
`sub_content_type`	string	This tag specifies the sub-content type to be associated with the output group. Either: • `default` • `SSPlayReady` • `HLSPlayReady` , or • Other customer-supported value.

SKE Key Server Source Settings

Use the following tags in an HLS Encryption (AES 128) filter when `key_source` is **`ske_key_server`**.

Tag	Type	Description
`service_url`	location	This tag specifies the location of the Irdeto server. Both a URL and login credentials are required. For contents of the location type, see Location Type Elements.
`account_id`	string	This is your account on the Irdeto Control server.
`content_id`	string	This tag identifies the Delta content in Irdeto Control. Each `content_id` value you assign in Delta should be unique for this Irdeto Control.

Tag	Type	Description
`username`	string	The username used to authenticate to the keyprovid er.
`password`	string	The password to authenticate to the keyprovider.

CA Key Server Source Settings

Use the following tags in an HLS Encryption (AES 128) filter when `key_source` is **`ca_key_server`**.

> **Important**
>
> This feature is beta in this release.

Tag	Type	Description
`service_url`	location	This tag specifies the location of the Irdeto server. Both a URL and login credentials are required. For contents of the location type, see Location Type Elements.
`account_id`	string	This is your account on the Irdeto Control server.
`content_id`	string	This tag identifies the Delta content in Irdeto Control. Each `content_id` value

Tag	Type	Description
		you assign in Delta should be unique for this Irdeto Control.
`username`	string	The username used to authenticate to the keyprovid er.
`password`	string	The password to authenticate to the keyprovider.
`kms_username`	string	The username for access to the Key Management System (KMS).
`kms_password`	string	The password for access to the Key Management System (KMS).

1Mainstream

The following table describes the tags that are specific for the 1Mainstream server.

Tag	Type	Description
`channel_secret`	string	The string that signs key requests sent to the 1Mainstream server.
`video_id`	string	The tag that identifies the Delta content to the 1Mainstream server. Each `video_id` you assign in Delta should be unique for this 1Mainstream server.
`channel_code`	string	The tag that identifies the Delta content to the

Tag	Type	Description
		1Mainstream server. Each `channel_value` you assign in Delta should be unique for this 1Mainstream server.
`base_url`	location	The location of the 1Mainstream server. Optional username and password are used if the keyprovider requires authentication. For contents of the location type, see Location Type Elements.

Nagra

The following table describes the tags that are specific for the Nagra server.

Tag	Type	Description
`content_id`	string	This tag identifies the Delta content on the Nagra server. Each `content_id` value you assign in Delta should be unique for this Nagra server.
`server_url`	string	This is the location of the Nagra server.

Piksel

The following table describes the tags that are specific for the Piksel server.

Tag	Type	Description
`content_id`	string	This tag identifies the Delta content on the Piksel server. Each `content_id` value you assign in Delta should be unique for this Piksel server.
`server`	location	This is the location of the Piksel server. Optional username and password are used if the keyprovider requires authentication. For contents of the location type, see Location Type Elements.

Secure Media

The following table describes the tags that are specific for the Secure Media server.

Tag	Type	Description
`resourceid`	integer	This tag is the Secure Media Resource ID. Must be between 0 – 4294967295.
`secure_media_server`	location	The Secure Media server that provides the keys. For contents of the location type, see Location Type Elements.
`reuse_last_key`	boolean	• **`false`** is the default and means that, if the secure media server becomes

Tag	Type	Description
		unreachable, the endpoint on this DRM filter and endpoints downstream of this DRM filter are not accessible. • **`true`** means that the output is encrypted using the last key obtained from the SecureMedia server if the secure media server becomes unreachable.

Self-Generated

The following table describes the tags that are specific for the self-generated server.

Tag	Type	Description
`key_prefix`	string	A partial URI prefix that is prepended to the key filenames in the output manifest. The prefix should point to the final publishing destination for the keys.
`common_key`	boolean	• **`true`** means that the same key is used for all bitrates associated with this content. • **`false`** means that different keys are generated for each bitrate.

Secure Package and Encoder Key Exchange (SPEKE)

Use the following tags in the `keyprovider_settings` object when using Secure Package and Encoder Key Exchange (SPEKE) for your keyprovider.

Tag	Type	Description
`system_ids`	string	This is the unique identifier for your DRM solution provider. For a list of common system IDs, see https://dashif.org/identifiers/content_protection/. If you don't know your ID, ask your DRM solution provider.
`content_id`	string	This identifies the Delta content on the Nagra server. Each `content_id` value you assign in Delta should be unique for this Widevine server.
`speke_server`	object	This holds information specific to the SPEKE server. See the following tables for tags for each server.

`speke_server` Settings

Include the following settings in the `speke_server` object.

Tag	Type	Description
`uri`	string	The location of the SPEKE server that provides your DRM keys.

Tag	Type	Description
`username`	string	The username when credentia ls are required to access a file or publishing point.
`password`	string	The password when credentia ls are required to access a file or publishing point.

Static

The following table shows the specific tags for the Static server.

Tag	Type	Description
`common_key`	string	The Fairplay key. This is a 128-bit hexadecimaladecima l value represented by a 32-character string.
`skd`	string	The Fairplay SKD/URI, formatted as **`skd://`**

Verimatrix

The following table shows the specific tags for the Verimatrix server.

Tag	Type	Description
`resourceid`	string	The Verimatrix Resource ID.
`verimatrix_server`	location	The Verimatrix server that provides the keys.

Tag	Type	Description
		For contents of the location type, see Location Type Elements.
`reuse_last_key`	boolean	• **`false`** is the default and means that, if the Verimatrix server becomes unreachable, the endpoint on this DRM filter and endpoints downstream of this DRM filter are not accessible. • **`true`** means that the output is encrypted using the last key obtained from the Verimatrix server if the server becomes unreachab le.
`resource_type`	string	• **`vod`** is for output VOD content or after Live to VOD filters. • **`dtv`** is for output live content.

VOSP

Enter the following settings in the `filter_settings` object.

Tag	Type	Description
`server_url`	string	The URL to query for the VOSP content key.
`service_id`	string	The VOSP Service ID.

Tag	Type	Description
`license_url`	string	The URL for the license acquisition web service.
`ui_license_url`	string	The URL for a non-silent license acquisition webpage.
`key_id`	string	A key ID. Must be a valid GUID.
`content_key_base64`	string	The tag that contains contains a base 64-encoded content key. See `content_key_hexadecimal` for details.
`content_key_hexadecimal`	string	The tag that contains contains a hexadecimaladecimal-encoded content key. This value is intended for testing purposes and allows you to use a key without contacting the server. To for testing, enter a value in this tag or in `content_key_base64`. Do not enter a value in both tags. To generate a content key by contacting the server, leave both these tags blank and enter a value in `server_url`.

Tag	Type	Description
custom_attributes	string	Custom information you want to attach. The downstream system does not act on this data.

PlayReady Output Filter

The following sections describe how to format the request to add a PlayReady output filter to a contents object.

Body of HTTP

The body of your request is XML content consisting of one `filters` container that holds one `filter` element. The `filter` element contains the following:

- One `filter_settings` element that holds several tags, including the:
  - `filter ID`
  - `keyprovider_type`
  - `keyprovider_settings` element that captures information about the keyprovider that you use. This element does not exist if the keyprovider is Custom, Microsoft, or VOSP. The tags for these keyproviders exist inside the `filter_settings` element rather than within the `keyprovider_settings`.

Request Example

This request modifies the existing content entity that has ID 39 to add a PlayReady output filter with Irdeto as the key provider type (specified about halfway down the body). In this example, `static_key` is set to **`false`**. See the HLS Encryption example (above) for sample values when `static_key` is **`true`**.

```
PUT http://10.24.34.2:8080/contents/39
  ---------------------------------------
  <content>
   <filters>
   <filter>
   <parent_id>419</parent_id>
   <filter_type>playready</filter_type>
```

```
    <endpoint>false</endpoint>
    <output_url/>
    <description/>
    <filter_settings>
    <initial_iv>1</initial_iv>
    <key_id>bae13835-7328-ey0a-91c7-ad4dbd8dbf8c<</key_id>
    <content_key>YY123uT5678==</content_key>
    <license_url>http://examplelicenseurl.com</license_url>
    <keyprovider_type>vosp</keyprovider_type>
    <server_url>http://serverurl/example</server_url>
    <service_id>your-service-id/service_id>
    <content_key_base64>N271RyR8dclFMQyfZWWxFA==</content_key_base64>
    <content_key_hexadecimal>3d9bc63659e69f57cff39b609c421631</
content_key_hexadecimal>
    </filter_settings>
    </filter>
    </filters>
   </content>
```

Topics

- The PlayReady filter Element
- The PlayReady filter_settings Element
- The PlayReady keyprovider_settings Element

The PlayReady `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only.

Tag	Type	Description
		Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only. The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. This customized address including the `output_url` is automatically create and stored in the `custom_endpoint_uri` tag.

Tag	Type	Description
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.

Tag	Type	Description
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The PlayReady `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

With the `filter_settings` for the Playready filter, a given tag may go in one of two places, depending on the keyprovider being used: for some keyproviders, the tag goes in the `filter_settings` element, while, for other keyproviders, it goes in the `keyprovider_settings` sub-element of `filter_settings`.

Tag	Type	Description
`initial_iv`	integer	The initial value of the initialization vector (IV). Default is **1**.
`keyprovider_type`	string	The Keyprovider types. Available options: • **`conax`** • **`custom`** (default) • **`irdeto`** • **`microsoft`** • **`piksel`** • **`speke`** • **`vosp`**
`keyprovider_settings`	object	The tag that holds the settings specific to your keyprovider. See The PlayReady `keyprovider_settings` Element for settings by keyprovider. Note that VOSP, Custom, and Microsoft information is captured in the `filter_settings` object.
`keyprovider_settings_id`	integer	The Delta-assigned identifier for the selected keyprovider.
`program_id`	string	The unique program ID identifier associated with a set of keys. It is passed to

Tag	Type	Description
		the keyprovider URI (`uri` parameter).
`username`	string	The username used to authenticate to the keyprovid er.
`password`	string	The password used to authenticate to the keyprovid er.

The PlayReady `keyprovider_settings` Element

The following sections describe the settings for each keyprovider.

Conax

The following table shows the tags specific to the Conax server.

Tag	Type	Description
`content_id`	string	This tag provides the Conax server with an ID to identify the Delta content. Each `content_id` value you assign in Delta should be unique for this Conax server.
`server`	location	This tag specifies the location of the Conax server. A URL and login credentials are required. For contents of the location type, see Location Type Elements.

Tag	Type	Description
`username`	string	The username used to authenticate to the keyprovider.
`password`	string	The password used to authenticate to the keyprovider.
`resource_type`	string	This is the Conax resource type. Use: • **`vod`** for VOD content or after Live to VOD filters. • **`dtv`** for output live content.

Custom Provider

The following table shows the tags specific to the Custom Provider server.

Enter the following settings in the `filter_settings` object.

Tag	Type	Description
`key_id`	string	A key ID. Must be a valid GUID.
`key_seed`	string	A base 64-encoded key seed. Required unless `content_key` is specified.
`content_key`	string	A base 64-encoded content key. Required unless `key_seed` is specified.
`license_url`	string	The URL for a license acquisition web service.

Tag	Type	Description
`ui_license_url`	string	The URL for a non-silent license acquisition webpage.
`custom_attributes`	string	Custom information you want to attach. The downstream system does not act on this data.

Irdeto

The following table shows the tags specific to the Irdeto server.

Tag	Type	Description
`service_url`	server	This tag specifies the location of the Irdeto server. Both a URL and login credentials are required. For contents of the server type, see Server Type Elements.
`account_id`	string	This is your account on the Irdeto Control server.
`content_id`	string	This tag identifies the Delta content in Irdeto Control. Each `content_id` value you assign in Delta should be unique for this Irdeto Control.
`username`	string	The username used to authenticate to the keyprovid er.

Tag	Type	Description
`password`	string	The password used to authenticate to the keyprovid er.
`use_https`	boolean	• **`false`** is the default and means that Delta uses basic HTTP when communicating with the License Acquisition URL. • **`true`** means that Delta uses HTTPS in requests to the License Acquisition URL.
`sub_content_type`	string	This tag specifies the sub-content type to be associated with the output group. Either: • `default` • `SSPlayReady` • `HLSPlayReady` , or • Other customer-supported value
`ui_license_url`	string	The URL for a non-silent license acquisition webpage.
`custom_attributes`	string	Custom information you want to attach. The downstream system does not act on this data.

Microsoft

The following table shows the tags specific to the Microsoft server.

Enter the following settings in the `filter_settings` object.

Tag	Type	Description
`uri`	string	The URI used for retrieving the keys from the keyprovid er.
`ui_license_url`	string	The URL for a non-silent license acquisition webpage.
`custom_attributes`	string	Custom information you want to attach. The downstream system does not act on this data.

Piksel

The following table shows the tags specific to the Piksel server.

Tag	Type	Description
`content_id`	string	This tag identifies the Delta content on the Piksel server. Each `content_id` value you assign in Delta should be unique for this Piksel server.
`server`	location	This is the location of the Piksel server. Optional username and password are used if the keyprovider requires authentication. For contents of the location type, see Location Type Elements.

Secure Package and Encoder Key Exchange (SPEKE)

Use the following tags in the `keyprovider_settings` object when using Secure Package and Encoder Key Exchange (SPEKE) for your keyprovider.

Tag	Type	Description
`system_ids`	string	This is the unique identifie r for your DRM solution provider. For a list of common system IDs, see https://dashif.org/identifiers/content_protection/. If you don't know your ID, ask your DRM solution provider.
`content_id`	string	This identifies the Delta content on the Nagra server. Each `content_id` value you assign in Delta should be unique for this Widevine server.
`speke_server`	object	This holds information specific to the SPEKE server. See below.

`speke_server` Settings

Include the following settings in the `speke_server` object.

Tag	Type	Description
`uri`	string	The location of the Secure Package and Encoder Key Exchange (SPEKE) server that provides your DRM keys.

Tag	Type	Description
`username`	string	The username when credentia ls are required to access a file or publishing point.
`password`	string	The password when credentia ls are required to access a file or publishing point.

VOSP

The following table shows the tags specific to the VOSP server.

Enter the following settings in the `filter_settings` object.

Tag	Type	Description
`server_url`	string	The URL to query for the VOSP content key.
`service_id`	string	The VOSP Service ID.
`license_url`	string	The URL for the license acquisition web service.
`ui_license_url`	string	The URL for a non-silent license acquisition webpage.
`key_id`	string	A key ID. Must be a valid GUID.
`content_key_base64`	string	This tag contains a base 64-encoded content key. See `content_key_hexade cimal` for details.
`content_key_hexade cimal`	string	This tag contains a hexadecim aladecimal-encoded content

Tag	Type	Description
		key. This value is intended for testing purposesand allows you to use a key without contacting the server. To for testing, enter a value in this tag or in `content_key_base64` . Do not enter a value in both tags. To generate a content key by contacting the server, leave both these tags blank and enter a value in `server_url` .
`custom_attributes`	string	Custom information you want to attach. The downstream system does not act on this data.

PUT: Create an Authentication Output Filter

Authentication output filters contain authentication information so that access to the content can be handled by an authentication mechanism. To create the output filter, send a PUT request for the content entity that corresponds to the content that authentication applies to. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID
```

Body of HTTP

The body of your request is XML content. The structure of the XML content depends on the type of filter (`filter_type` tag in each `filter` element in the body).

See the following sections for information about each authentication output filter.

- Akamai G2o Output Filter
- Cisco URL Signing Output Filter

Response

The response repeats back all possible tags in the filter you created (including those you did not set) with the addition of:

- `id`: Unique IDs for the filter and each sub-element as described in this section.
- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI as follows:

```
http://Delta IP address:8080/out/i/filter ID.extension
```

  where:

  - `filter ID` is the ID for this filter, assigned once the filter is created.
  - `extension` is different for each output filter type.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if the `output_uri` tag contains a value. Contains the custom endpoint URI, as follows:

```
http://Delta IP address:8080/out/u/output_url.m3u8
```

See the following sections for information about and example requests for each authentication output filter.

Akamai G2o Output Filter

The following sections describe how to format the request to add an Akamai G2o output filter to a contents object.

Body of HTTP

The body of your request is XML content consisting of one `filters` container that holds one `filter` element. The `filter` element contains the following:

- One `filter_settings` element that holds several tags and also has:
  - Zero or one `nonce_key_pairs` container and one or more `nonce_key_pair` elements.

Request Example

This request creates an Akamai G2o filter in the content entity with the ID 60. It adds the filter after the existing filter that has the ID 40. It uses some default values (so the tags are not specified), but it explicitly creates one `nonce_key_pair`.

```
PUT http://10.24.34.2:8080/contents/60
-----------------------------------------------
<content>
    <filters>
        <filter>
            <parent_id>40</parent_id>
            <endpoint>false</endpoint>
            <filter_type>akamai_g2o_authentication</filter_type>
            <filter_settings>
                <ip_ranges>
                    <ip_range>
                    <ip_range>10.24.34.20/24</ip_range>
                    <ip_range>
                </ip_ranges>
                <nonce_key_pairs>
                    <nonce_key_pair>
                        <nonce>123456</nonce>
                        <key>5c81fbf7cbcbc6efff2d79e4</key>
                    </nonce_key_pair>
                </nonce_key_pairs>
            </filter_settings>
        </filter>
    </filters>
</content>
```

Response

The entire content entity is returned. All the default values for the `filter_settings` are shown. The new filter and its `filter_settings` element are assigned unique IDs.

```
<?xml version="1.0" encoding="UTF-8"?>
<filters href="/contents/60/filters" product="Delta" version="2.3.0.123456">
    <filter href="/contents/60/filters/40" product="Delta" version="2.3.0.123456">
        <id>40</id>
        .
        .
```

```
        .
    </filter>
    <filter href="/contents/60/filters/2130" product="Delta" version="2.3.0.123456">
        <id>43</id>
        <parent_id>40</parent_id>
        <endpoint>false</endpoint>
        <output_url/>
        <filter_type>akamai_g2o_authentication</filter_type>
        <filter_settings>
            <id>3</id>
            <ip_ranges>
                <ip_range>
                    <ip_range>10.24.34.20/24</ip_range>
                <ip_range>
            </ip_ranges>
            <clear_data_header_name>X-Akamai-G2O-Auth-Data</clear_data_header_name>
            <encrypted_data_header_name>X-Akamai-G2O-Auth-Sign</
encrypted_data_header_name>
            <request_validity_window>30</request_validity_window>
            <nonce_key_pairs>
                <nonce_key_pair>
                    <nonce>123456</nonce>
                    <key>5c81fbf7cbcbc6efff2d79e4</key>
                </nonce_key_pair>
            </nonce_key_pairs>
        </filter_settings>
    </filter>
</filters>
```

Topics

- The Akamai G2O Authentication filter Element
- The Akamai G2O Authentication filter_settings Element

The Akamai G2O Authentication `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only. The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. This customized address

Tag	Type	Description
		including the `output_url` is automatically create and stored in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.

Tag	Type	Description
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The Akamai G2O Authentication `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
`clear_data_header_name`	string	The HTTP header containing clear G2O settings. Default: X-Akamai-G2O-Auth-Data

Tag	Type	Description
`encrypted_data_header_name`	string	The HTTP header containing G2O signature. Default: X-Akamai-G2O-Auth-Data
`request_validity_window`	integer	The amount of time in seconds for the server to consider a G2O request valid. Default is **`30`**. It allows you to correct for time differences between the Delta node and Akamai edge servers.

Tag	Type	Description
`nonce_key_pairs`	object	A list of one or more `nonce_key_pair` elements. Each nonce_key_pair consists of: • `nonce`: A key used for generating the signature. • `key`: Encryption key used for generating the signature. The `nonce_key_pair` is used for signature validatio n. These must be set to equal the nonce and key pair in the Akamai edge configuration. If the edge configuration changes, a new `nonce_key _pair` should be added and the previous one retained until transition to the new key is complete.
`ip_ranges`	object	A list of `ip_address` tags for client players (or other downstream requester) that can bypass Akamai G2O authentication. Can be either: • A single static IP address, or • A range of IP addresses in CIDR notation

Cisco URL Signing Output Filter

The following sections describe how to format the request to add a Cisco URL Signing output filter to a contents object.

Body of HTTP

The body of your request is XML content consisting of one `filters` container that holds one `filter` element. The `filter` element contains the following:

- One `filter_settings` element that holds several tags.

Request Example

This request creates a Cisco URL Signing filter in the content entity with the ID 20. It adds the filter after the existing filter that has the ID 23.

```
PUT http://10.24.34.2:8080/contents/20
-----------------------------------------------
<content>
    <filters>
        <filter>
            <parent_id>23</parent_id>
            <endpoint>false</endpoint>
            <output_url></output_url>
            <filter_type>cisco_url_signing</filter_type>
            <filter_settings>
                <url_prefix>http://domain.mycdn01.com</url_prefix>
                <client_ip>1.2.3.4</client_ip>
                <expiration_window>10800</expiration_window>
                <key_owner_id>1</key_owner_id>
                <key_number_id>11</key_number_id>
                <key>F2e9eueuis8Vh3Yx</key>
                <exclude_domain>false</exclude_domain>
                <version>2</version>
            </filter_settings>
        </filter>
    </filters>
</content>
```

Response

The entire content entity is returned. All the default values for the `filter_settings` are shown. The new filter and its `filter_settings` element are assigned unique IDs.

```
<?xml version="1.0" encoding="UTF-8"?>
<filters href="/contents/20/filters" product="Delta" version="2.3.0.123456">
    <filter href="/contents/20/filters/33" product="Delta" version="2.3.0.123456">
        <id>76</id>
        <parent_id>23</parent_id>
        <endpoint>false</endpoint>
        <output_url></output_url>
        <filter_type>cisco_url_signing</filter_type>
        <filter_settings>
            <id>64</id>
            <url_prefix>http://domain.mycdn01.com</url_prefix>
            <client_ip>1.2.3.4</client_ip>
            <expiration_window>10800</expiration_window>
            <key_owner_id>1</key_owner_id>
            <key_number_id>11</key_number_id>
            <key>F2e9eueuis8Vh3Yx</key>
            <exclude_domain>false</exclude_domain>
            <version>2</version>
        </filter_settings>
    </filter>
</filters>
```

Topics

- The Cisco URL Signing filter Element
- The Cisco URL Signing filter_settings Element

The Cisco URL Signing `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.

Tag	Type	Description
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only. The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. This customized address including the `output_url` is automatically create and stored in the `custom_endpoint_uri` tag.

Tag	Type	Description
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.

Tag	Type	Description
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The Cisco URL Signing `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
`url_prefix`	string	Prefix of URL to be signed. This should correspond to the public URL from which the content is accessed.

Tag	Type	Description
`client_ip`	string	Client IP address to generate signature for. • If the CDS is configured to require client IP validation, enter a valid IP address. • If the CDS has disabled client IP validation, you must still enter an IP address: enter **`1.2.3.4`**.
`expiration_window`	integer	• If the Internet Streaming CDS is configured to validate expiration dates, enter a number of seconds. If this number of seconds pass between the time the URL signature is created and the time the CDS validates the request, then the validation fails. • If the Internet Streaming CDS is configured to ignore expiration date (via the `exclude_validation` attribute), then enter any number.

Tag	Type	Description
`key_owner_id`	integer	The ID number for the owner of the encryption key specified in the key tag. The Internet Streamer CDS must already be configured with this key owner ID. Range: 1-32
`key_number_id`	integer	The Key ID number for the encryption key specified in the key tag. The Internet Streamer CDS must already be configured with this key ID number. Range: 1-16
`key`	string	The Key that is used to generate the URL signature s. The Internet Streamer CDS must already be configured with this key.
`exclude_domain`	boolean	• **`true`** means that the CDS is configured to exclude the domain. The domain is omitted from the URL signature. • **`false`** means that the CDS is configured to include the domain. The domain is included in the URL signature.

Tag	Type	Description
`version`	integer	Version of signature algorithm. • **0**: MD5 hash algorithm. • **1**: SHA-1 hash algorithm. • **2**: SHA-1 hash algorithm with the protocol removed from the beginning of the URL (without schema).

PUT: Create a Whitelist or Blacklist Output Filter

Whitelist and Blacklist output filters restrict what IP addresses can receive content from the endpoint. To create the output filter, send a PUT request for the content entity that corresponds to the content that the whitelist or blacklist applies to. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID
```

Body of HTTP

The body of your request is XML content consisting of one `filters` container that holds one `filter` element. The `filter` element contains several tags and also holds the following:

- One `filter_settings` element that contains one tag.

Request Example

This request modifies the existing content entity that has ID 39 to add an IP whitelist filter after the output filter with the ID 13. The whitelist allows only players with an IP address in the range that starts with 10.10.4.400 and ends with 10.10.5.300.

```
PUT http://10.24.34.2:8080/contents/39
---------------------------------------
<content>
    <filters>
```

```
        <filter>
            <parent_id>13</parent_id>
            <endpoint>true</endpoint>
            <output_url/>
            <filter_type>ip_whitelist</filter_type>
            <filter_settings>
                <ip_ranges>
                    <ip_range>
                        <ip_range>10.24.34.20/24</ip_range>
                    <ip_range>
                </ip_ranges>
            </filter_settings>
        </filter>
    </filters>
</content>
```

Response

The response repeats back all the possible tags in the filter you created (including those you did not set) with the addition of:

- `id`: Unique IDs for the filter and each sub-element as described in this section.
- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI as follows:

  ```
  http://Delta IP address:8080/out/i/filter ID.<extension>
  ```

  where:

  - `filter ID` is the ID for this filter, assigned once the filter is created.
  - `extension` is different for each output filter type.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if the `output_uri` tag contains a value. Contains the custom endpoint URI, as follows:

  ```
  http://Delta IP address:8080/out/u/output_url.m3u8
  ```

The response is identical to the response to a Viewing Filters List: GET Output Filters List.

Topics

- The IP Whitelist and IB Blacklist filter Element

- The IP Whitelist and IP Blacklist filter_settings Element

The IP Whitelist and IB Blacklist `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only.

Tag	Type	Description
		The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. This customized address including the `output_url` is automatically create and stored in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.

Tag	Type	Description
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The IP Whitelist and IP Blacklist `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
`ip_range`	string	For IP Whitelist: • The IP address of the single client player that can access the content.

Tag	Type	Description
		• A range of IP addresses in CIDR notation that can access the content. All of the IP addresses in the range (including the first and last) is able to access the content. For IP Blacklist: • The IP address of the single client player that cannot access the content. • A range of IP addresses in CIDR notation that cannot access the content. None of the IP addresses in the range (including the first and last) is able to access the content.

PUT: Create a User Agent Output Filter

User Agent output filters makes it so only specific types of users have access to content. To create the output filter, send a PUT request for the content entity that corresponds to the content that the user agents apply to. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID
```

Body of HTTP

The body of your request is XML content consisting of one `filters` container that holds one `filter` element. The `filter` element contains several tags and also holds the following:

- One `filter_settings` element that contains one tag.

Request Example

This request modifies the existing content entity that has ID 39 to add a user agent filter after the output filter that has the ID 13. The filter specifies one user agent preset (ID 2).

```
PUT http://10.24.34.2:8080/contents/39
  ---------------------------------------
   <content>
   <filters>
   <filter>
   <parent_id>13</parent_id>
   <endpoint>true</endpoint>
   <output_url/>
   <filter_type>user_agent</filter_type>
   <filter_settings>
   <user_agent_preset_ids>2</user_agent_preset_ids>
   </filter_settings>
   </filter>
   </filters>
   </content>
```

Response

The response repeats back all the possible tags in the filter you created (including those you did not set) with the addition of:

- `id`: Unique IDs for the filter and each sub-element as described in this section.
- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI as follows:

  ```
  http://Delta IP address:8080/out/i/filter ID.<extension>
  ```

  where:

  - `filter ID` is the ID for this filter, assigned once the filter is created.
  - `extension` is different for each output filter type.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if the `output_uri` tag contains a value. Contains the custom endpoint URI as follows:

  ```
  http://Delta IP address:8080/out/u/output_url.m3u8
  ```

This example response is a representation. The newly created user agent filter has ID 6.

```
<?xml version="1.0" encoding="UTF-8"?>
  <contents href="/contents" product="Delta" version="2.3.0.123456">
   <content href="/contents/39" product="Delta" version="2.3.0.123456">
   <id>39</id>
   <name>comedy_2.m3u8</name>
   <path></path>
   <path>/data/server/mount/content/comedy_2.m3u8</path>
   <type>HLS</type>
   <remote_input>false</remote_input>
   <filters>
   <filter>
   .
   .
   </filter>
   <filter href="/contents/39/filters/6" product="Delta" version="2.3.0.123456">
   <id>6</id>
   <parent_id>13</parent_id>
   <endpoint>true</endpoint>
   <default_endpoint_uri>http://10.24.34.2/out/i/6.m3u8</default_endpoint_uri>
   <custom_endpoint_uri></custom_endpoint_uri>
   <output_url/>
   <filter_type>user_agent</filter_type>
   <filter_settings>
   <id>2</id>
   <user_agent_preset_ids>2</user_agent_preset_ids>
   </filter_settings>
   </filter>
   </filters>
   </content>
  </contents>
```

Topics

- The User Agent filter Element
- The User Agent filter_settings Element

The User Agent `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only.

Tag	Type	Description
		The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. This customized address including the `output_url` is automatically create and stored in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.

Tag	Type	Description
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The User Agent `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
`user_agent_preset_ids`	string	Either: • The numerical ID of one existing user preset, or

Tag	Type	Description
		• A comma-separated list of numerical IDs. See GET User Agent Presets List for information on obtaining a list of existing user presets. Each user agent preset is a regular expression that identifies valid values for the user agent data in the request from the device (user agent) to Delta. When a request from a device matches this regex, Delta declares a match on the branch of the filter tree where this user agent filter is located and delivers the content represented by this branch.

Working with User Agent Presets

The user agent filter contains a `user_agent_preset` tag. These presets are encapsulated in the `user_agent_presets` container.

See the following sections for actions you can take against user agent presets.

Topics

- POST User Agent Preset
- GET User Agent Presets List
- DELETE User Agent Preset

POST User Agent Preset

To create a user agent preset, send a POST request. The following sections describe how to format the request.

HTTP URL

```
POST HTTP://Delta IP address:8080/user_agent_presets
```

Body of HTTP

The body of your request is XML content consisting of one `user_agent_preset` element that contains the following:

Tag	Type	Description
`name`	string	Name for the preset: must be unique among all presets.
`regex`	string	The user agent value, expressed as a regex. When a request from a user agent (device) matches this regex, Delta makes a match on this branch of the filter tree and delivers the content represented by this branch.

Request Example

```
POST http://10.24.34.2:8080/user_agent_presets
-------------------------------------------------------
<user_agent_preset>
    <name>iOS no iPod</name>
    <regex>(iPad)|(iPhone)</regex>
</user_agent_preset>
```

Response

The response repeats back all the possible tags in the filter you created (including those you did not set) with the addition of:

- An HREF in the `user_agent_preset` element. This HREF includes a unique ID for the preset.

```
<?xml version="1.0" encoding="UTF-8"?>
<user_agent_preset href="/user_agent_presets/10" product="Delta"
 version="2.3.0.123456">
    <name>iOS no iPod</name>
    <regex>(iPad)|(iPhone)</regex>
</user_agent_preset>
```

GET User Agent Presets List

To retrieve a list of user agent presets that are in Delta, send a GET request. The following sections describe how to format the request.

HTTP URL

```
GET HTTP://Delta IP address:8080/user_agent_presets
```

Request Example

```
GET HTTP://10.24.34.2:8080/user_agent_presets
```

Response

XML content consisting of one `user_agent_presets` container that holds zero or more `user_agent_preset` elements, each holding:

- An HREF that includes a unique ID for the preset.
- Use the following tags to obtain user presets:

Tag	Type	Description
name	string	Name for the preset: must be unique among all presets.

Tag	Type	Description
regex	string	The user agent value, expressed as a regular expression to allow matching on different possible values.

In this example, the first nine presets are those that are built into Delta. The last is one that was created using a POST request.

```
<?xml version="1.0" encoding="UTF-8"?>
<user_agent_presets>
    <user_agent_preset href="/user_agent_presets/1" product="Delta"
 version="2.3.0.123456">
        <name>Android</name>
        <regex>Android</regex>
    </user_agent_preset>
    <user_agent_preset href="/user_agent_presets/2" product="Delta"
 version="2.3.0.123456">
        <name>Chrome</name>
        <regex>Chrome</regex>
    </user_agent_preset>
    <user_agent_preset href="/user_agent_presets/3" product="Delta"
 version="2.3.0.123456">
        <name>Firefox</name>
        <regex>Firefox</regex>
    </user_agent_preset>
    <user_agent_preset href="/user_agent_presets/4" product="Delta"
 version="2.3.0.123456">
        <name>Internet Explorer</name>
        <regex>(MSIE)|(Trident)</regex>
    </user_agent_preset>
    <user_agent_preset href="/user_agent_presets/5" product="Delta"
 version="2.3.0.123456">
        <name>iOS</name>
        <regex>(iPad)|(iPod)|(iPhone)</regex>
    </user_agent_preset>
    <user_agent_preset href="/user_agent_presets/6" product="Delta"
 version="2.3.0.123456">
        <name>iPad</name>
        <regex>iPad</regex>
```

```
    </user_agent_preset>
    <user_agent_preset href="/user_agent_presets/7" product="Delta"
 version="2.3.0.123456">
        <name>iPhone</name>
        <regex>iPhone</regex>
    </user_agent_preset>
    <user_agent_preset href="/user_agent_presets/8" product="Delta"
 version="2.3.0.123456">
        <name>Macintosh</name>
        <regex>Macintosh</regex>
    </user_agent_preset>
    <user_agent_preset href="/user_agent_presets/9" product="Delta"
 version="2.3.0.123456">
        <name>Windows</name>
        <regex>Windows</regex>
    </user_agent_preset>
    <user_agent_preset href="/user_agent_presets/10" product="Delta"
 version="2.3.0.123456">
        <name>iOS no iPod</name>
        <regex>(iPad)|(iPhone)</regex>
    </user_agent_preset>
</user_agent_presets>
```

DELETE User Agent Preset

To remove a user agent preset from Delta, send a DELETE request.

> **Note**
>
> You can delete any of the built-in presets but we strongly recommend that you do not do so.

HTTP URL

```
DELETE HTTP://Delta IP address:8080/user_agent_presets/ID of preset
```

Response

A 200 OK response indicates the delete was successful.

Creating Processing Output Filters in AWS Elemental Delta

Processing output filters process the output content in various ways.

Topics

- PUT: Create an Ad Removal Output Filter
- PUT: Create an Ad Replace Output Filter
- PUT: Create a Bitrate Selector Output Filter
- PUT: Create a Blackout Output Filter
- PUT: Create a Cache Settings Output Filter
- PUT: Create a File Copy Output Filter
- PUT: Create a Live to VOD Output Filter
- PUT: Create a Live to VOD Catalog Output Filter
- PUT: Create a Preroll Ad Insertion or Postroll Ad Insertion Output Filter
- PUT: Create a Time Delay Output Filter
- PUT: Create a VOD Clip Output Filter

PUT: Create an Ad Removal Output Filter

The Ad Removal output filter removes all ad content from the media stream and takes out all ad markers from the manifest. To create the output filter, send a PUT request for the content object that corresponds to the content that you're removing ads from. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID/filters
```

Body of HTTP

The body of your request is XML content consisting of one `filters` container that holds one `filter` element with several tags, as described in of The Ad Removal `filter` Element.

Request Example

See the Live to VOD Request Example for a general processing output filter request.

Response

The response repeats back the filter that you created, with the addition of the following tags:

- `id`: Unique IDs for the filter and each sub-element as described in this section.
- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI, as follows:

  ```
  http://<Delta IP address>:8080/out/i/<filter ID>.<extension>
  ```

  where `filter ID` is the ID for this filter, assigned once the filter is created.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if `output_uri` contains a value. Contains the custom endpoint URI, as follows:

  ```
  http://<Delta IP address:8080>/out/u/<output_url>.<extension>
  ```

This example response is a representation. The `default_endpoint` now shows a value. The `filter_settings` shows the default values. The `filter`, `filter_settings`, `stream_set`, `video_track`, and `audio_track` elements are all assigned unique IDs. Your results may vary.

Topics

- of The Ad Removal filter Element
- The Ad Removal filter_settings Element

of The Ad Removal `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only.

Tag	Type	Description
		Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only. The Delta-assigned default playback endpoint for the output filter.

Tag	Type	Description
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. This customized address including the `output_url` is automatically create and stored in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url`.
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.

Tag	Type	Description
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The Ad Removal `filter_settings` Element

There are no `filter_settings` for `ad_removal` filters.

PUT: Create an Ad Replace Output Filter

The Ad Replace output filter replaces existing content with ad content that's stored on Delta or from a VAST server. To create the output filter, send a PUT request for the content object that corresponds to the content that you're placing ads in. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID/filters
```

Body of HTTP

The body of your request is XML content consisting of one `filters` container that holds one `filter` element with several tags, and:

- One `filter_settings` element that holds one tag. See the section called "The Ad Replace `filter_settings` Element".

Request Example

See the Live to VOD Request Example for a general processing output filter request.

Response

The response repeats back the filter that you created, with the addition of the following tags:

- `id`: Unique IDs for the filter and each sub-element as described in this section.
- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI, as follows:

  ```
  http://<Delta IP address>:8080/out/i/<filter ID>.<extension>
  ```

  where `filter ID` is the ID for this filter, assigned once the filter is created.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if `output_uri` contains a value. Contains the custom endpoint URI, as follows:

  ```
  http://<Delta IP address:8080>/out/u/<output_url>.<extension>
  ```

This example response is a representation. The `default_endpoint` now shows a value. The `filter_settings` shows the default values. The `filter`, `filter_settings`, `stream_set`, `video_track`, and `audio_track` elements are all assigned unique IDs. Your results may vary.

Topics

- The Ad Replace filterElement
- The Ad Replace filter_settings Element

The Ad Replace `filterElement`

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only.

Tag	Type	Description
		The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. This customized address including the `output_url` is automatically create and stored in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.

Tag	Type	Description
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The Ad Replace `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
ad_mode	string	This tag describes what action Delta takes with ad content. • **`insert`**: VOD content only. Ads are inserted in predetermined ad slots.

Tag	Type	Description
		• **`replace`**: Live or VOD content. Ads replace main content.
`ad_source_type`	string	This tag identifies the source of the ad content. • **`static`**: The ad content is Delta content. • **`vast`**: The ad content is being provided by a VAST server.
`ad_source_settings`	object	This object holds tags that describe the source of the ads. For available tags, see the following table.

The `ad_source_settings` Element

The `ad_source_settings` object describes where you store the ads that Delta uses for pre-roll and post-roll ad insertion. The following table describes the tags available for `ad_source_settings`.

Tag	Type	Description
`ad_sources`	string	This tag provides informati on about the source of the ads, based on the value of `ad_source_type` . • If `ad_source_type` is **`static`**: Enter a comma-separated list of the IDs (from the `id` tag) or aliases (from the `alias` tag) of the

Tag	Type	Description
		Delta content object to use as ad content. • If `ad_source_type` is **`vast`**: Enter a list of ad content tags (for static content) that Delta can use if it can't reach the VAST server. Advertising content should be VOD content-formatted to match ad avail timing. For example, if ad avails are 60 seconds, ad content should be in 15-, 30-, or 60-second intervals. Delta moves through the list of content as ad avails appear in the stream. When all specified sources have been inserted into the stream once, Delta restarts at the beginning of the list.
`campaign_id`	integer	If applicable, the ad campaign ID that you use with the VAST server. This ID is valid only if `ad_source_type` is **`vast`**, in which case it is optional.

Tag	Type	Description
`server`	location	The location of the VAST server. This value is required only if `ad_source_type` is **`vast`**. For contents of the location type, see Location Type Elements.

PUT: Create a Bitrate Selector Output Filter

The Bitrate Selector output filter is an advanced filter that temporarily restricts the bitrates that Delta serves. To create the output filter, send a PUT request for the content object that corresponds to the content that you're adding restrictions to. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID/filters
```

Body of HTTP

The body of your request is XML content consisting of one `filters` container that holds one `filter` element with several tags and the following:

- One `filter_settings` element that contains one tag.
- One or more `selected_bitrate` elements. See the section called "The Bitrate Selector `filter_settings` Element".

Request Example

See the Live to VOD Request Example for a general processing output filter request.

Response

The response repeats back the filter that you created, with the addition of the following tags:

- `id`: Unique IDs for the filter and each sub-element as described in this section.

- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI, as follows:

  ```
  http://<Delta IP address>:8080/out/i/<filter ID>.<extension>
  ```

  where `filter ID` is the ID for this filter, assigned once the filter is created.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if `output_uri` contains a value. Contains the custom endpoint URI, as follows:

  ```
  http://<Delta IP address:8080>/out/u/<output_url>.<extension>
  ```

This example response is a representation. The `default_endpoint` now shows a value. The `filter_settings` shows the default values. The `filter`, `filter_settings`, `stream_set`, `video_track`, and `audio_track` elements are all assigned unique IDs. Your results may vary.

Topics

- The Bitrate Selector filter Element
- The Bitrate Selector filter_settings Element

The Bitrate Selector `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.

Tag	Type	Description
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only. The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. This customized address including the `output_url` is automatically create and stored in the `custom_endpoint_uri` tag.

Tag	Type	Description
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.

Tag	Type	Description
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The Bitrate Selector `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

The `filter_settings` object holds one `selected_bitrates` container and one or more `selected_bitrate` elements, as described in the following table. When this filter is created, Delta serves only the specified bitrates for the endpoint enabled by this filter (if any) and for all filters downstream (that have an endpoint enabled).

Tag	Type	Description
`bitrate`	integer	Bitrate to be enabled or disabled. If a bitrate is not included in the list, it is disabled by default.
`enabled`	boolean	If **`enabled`**, then the bitrate is available if the Bitrate Selector Filter is an endpoint.

PUT: Create a Blackout Output Filter

The Blackout output filter replaces a portion of existing content with content stored on Delta. To create the output filter, send a PUT request for the content object that corresponds to the asset with content that you're replacing. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID/filters
```

Body of HTTP

The body of your request is XML content consisting of one `filters` container that holds one `filter` element with several tags and the following:

- One `filter_settings` element that contains one tag. See the section called "The Blackout `filter_settings` Element".

Request Example

See the Live to VOD Request Example for a general processing output filter request.

Response

The response repeats back the filter that you created, with the addition of the following tags:

- `id`: Unique IDs for the filter and each sub-element as described in this section.

- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI, as follows:

  ```
  http://<Delta IP address>:8080/out/i/<filter ID>.<extension>
  ```

  where `filter ID` is the ID for this filter, assigned once the filter is created.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if `output_uri` contains a value. Contains the custom endpoint URI, as follows:

  ```
  http://<Delta IP address:8080>/out/u/<output_url>.<extension>
  ```

This example response is a representation. The `default_endpoint` now shows a value. The `filter_settings` shows the default values. The `filter`, `filter_settings`, `stream_set`, `video_track`, and `audio_track` elements are all assigned unique IDs. Your results may vary.

Topics

- The Blackout filter Element
- The Blackout filter_settings Element

The Blackout `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.

Tag	Type	Description
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only. The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. This customized address including the `output_url` is automatically create and stored in the `custom_endpoint_uri` tag.

Tag	Type	Description
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.

Tag	Type	Description
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The Blackout `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
`content_ids`	string	A comma-separated list of Content IDs. These IDs identify standard Delta content that has been ingested via any input filter.

Tag	Type	Description
enabled	boolean	• **false** is the default and means that blackout is not enabled. Any network cues in the stream remain in the stream. • **true** means that blackout is enabled.

PUT: Create a Cache Settings Output Filter

The Cache Settings output filter modifies the cache time for output content by overriding all of the max-age HTTP headers for all endpoints in this filter chain. To create the output filter, send a PUT request for the content object that you're modifying cache settings for. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID/filters
```

Body of HTTP

The body of your request is XML content consisting of one `filters` container that holds one `filter` element with several tags and the following:

- One `filter_settings` element that contains one tag. See the section called "The Cache Settings `filter_settings` Element".

Request Example

See the Live to VOD Request Example for a general processing output filter request.

Response

The response repeats back the filter that you created, with the addition of the following tags:

- `id`: Unique IDs for the filter and each sub-element as described in this section.
- `default_endpoint_uri`: Included if `endpoint` is **true**. Contains the default endpoint URI, as follows:

```
http://<Delta IP address>:8080/out/i/<filter ID>.<extension>
```

where `filter ID` is the ID for this filter, assigned once the filter is created.

- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if `output_uri` contains a value. Contains the custom endpoint URI, as follows:

```
http://<Delta IP address:8080>/out/u/<output_url>.<extension>
```

This example response is a representation. The `default_endpoint` now shows a value. The `filter_settings` shows the default values. The `filter`, `filter_settings`, `stream_set`, `video_track`, and `audio_track` elements are all assigned unique IDs. Your results may vary.

Topics

- The Cache Settings filter Element
- The Cache Settings filter_settings Element

The Cache Settings `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag.

Tag	Type	Description
		To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only. The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. This customized address including the `output_url` is automatically create and stored in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .

Tag	Type	Description
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).

Tag	Type	Description
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The Cache Settings `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
`max_age_vod_variant_manifest`	integer	The amount of time (in seconds) that a downstream server should cache the VOD variant manifest. This applies to top-level HLS and HDS manifests for VOD and live-to-VOD content. Default 21600.
`max_age_vod_bitrate_manifest`	integer	The amount of time (in seconds) that a downstream server should cache the VOD bitrate manifest. This applies to top-level MSS and DASH manifests and HLS and HDS bitrate manifests for VOD and live-to-VOD content. Default 21600.
`max_age_live_variant_manifest`	integer	The amount of time (in seconds) that a downstream server should cache the live variant manifest. This applies to top-level HLS and HDS manifests for live content. Default is 2.

Tag	Type	Description
`max_age_live_bitrate_manifest`	integer	The amount of time (in seconds) that a downstream server should cache the live bitrate manifest. This applies to top-level MSS and DASH manifests and HLS and HDS bitrate manifests for live content. Default is 2.
`max_age_content_files`	integer	The amount of time (in seconds) that a downstream server should cache content files. Default is 21600.

PUT: Create a File Copy Output Filter

The File Copy output filter copies packaged content to a directory that you specify. To create the output filter, send a PUT request for the content object that corresponds to the asset that you're copying. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID/filters
```

Body of HTTP

The body of your request is XML content consisting of one `filters` container that holds one `filter` element with several tags and the following:

- One `filter_settings` element that contains one tag. See the section called “The File Copy `filter_settings` Element”.

Request Example

See the Live to VOD Request Example for a general processing output filter request.

Response

The response repeats back the filter that you created, with the addition of the following tags:

- `id`: Unique IDs for the filter and each sub-element as described in this section.
- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI, as follows:

```
http://<Delta IP address>:8080/out/i/<filter ID>.<extension>
```

  where `filter ID` is the ID for this filter, assigned once the filter is created.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if `output_uri` contains a value. Contains the custom endpoint URI, as follows:

```
http://<Delta IP address:8080>/out/u/<output_url>.<extension>
```

This example response is a representation. The `default_endpoint` now shows a value. The `filter_settings` shows the default values. The `filter`, `filter_settings`, `stream_set`, `video_track`, and `audio_track` elements are all assigned unique IDs. Your results may vary.

Topics

- The File Copy filter Element
- The File Copy filter_settings Element

The File Copy `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only.

Tag	Type	Description
		Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only. The Delta-assigned default playback endpoint for the output filter.

Tag	Type	Description
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. This customized address including the `output_url` is automatically create and stored in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.

Tag	Type	Description
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The File Copy `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
`path`	string	The location and name for the saved files: • **`<directory>/`** : files are named with the filter ID and saved in this directory.

Tag	Type	Description
		The directory is created if it does not yet exist. • **`<directory>/<name>`** : files are named with <name> and saved in this directory. The directory or directories are created if they do not yet exist. The directory must be accessible to Delta (for example, it must be a remote server mounted onto Delta via Settings > Mount Points). If you're using an S3 directory , the path must start with **`s3://`** and you must include `aws_credential_id` .
`aws_credential_id`	integer	The ID corresponding to the AWS credentials that this output filter uses to access Amazon S3. Required if you are copying files to an S3 bucket.

PUT: Create a Live to VOD Output Filter

The Live to VOD output filter creates a live-to-VOD (video on demand) asset from a portion of a live stream. To create the output filter, send a PUT request for the content object that corresponds to the live asset. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID/filters
```

Body of HTTP

The body of your request is XML content consisting of one `filters` container that holds one `filter` element with several tags and the following:

- One `filter_settings` element that contains one tag. See the section called "The Live to VOD `filter_settings`Element".

Request Example

This request modifies the content entity with the ID 21 to add a Live to VOD filter to capture the stream between 14:42:00 and 15:48:00 on June 4, 2015. The offset for the time is -7 hours, indicating the Pacific time zone.

```
PUT http://10.24.34.2:8080/contents/21
  ----------------------------------------
  <content>
   <filters>
   <filter>
   <filter_type>live_to_vod</filter_type>
   <endpoint>true</endpoint>
   <output_url/>
   <parent_id>13</parent_id>
   <filter_settings>
   <start_time>2017-11-27T16:03:47-08:00</start_time>
   <end_time>2017-11-27T16:05:47-08:00</end_time>
   <start_over>false</start_over>
   <manifest_scope>default</manifest_scope>
   <allow_url_start_end_params>false</allow_url_start_end_params>
   <start_frame/>
   <end_frame/>
   <frame_accurate>false</frame_accurate>
   <lowest_framerate_numerator>30000</lowest_framerate_numerator>
   <lowest_framerate_denominator>1001</lowest_framerate_denominator>
   </filter_settings>
   </filter>
   </filters>
  </content>
```

Response

The response repeats back the filter that you created, with the addition of the following tags:

- `id`: Unique IDs for the filter and each sub-element as described in this section.
- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI, as follows:

  ```
  http://<Delta IP address>:8080/out/i/<filter ID>.<extension>
  ```

  where `filter ID` is the ID for this filter, assigned once the filter is created.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if `output_uri` contains a value. Contains the custom endpoint URI, as follows:

  ```
  http://<Delta IP address:8080>/out/u/<output_url>.<extension>
  ```

This example response is a representation. The `default_endpoint` now shows a value. The `filter_settings` shows the default values. The `filter`, `filter_settings`, `stream_set`, `video_track`, and `audio_track` elements are all assigned unique IDs. Your results may vary.

```
<?xml version="1.0" encoding="UTF-8"?>
  <filter href="/contents/39/filters/21" product="Delta" version="2.3.0.123456">
   <id>21</id>
   <parent_id>13</parent_id>
   <endpoint>false</endpoint>
   <output_url/>
   <filter_type>live_to_vod</filter_type>
   <filter_settings>
   <id>1</id>
   <start_time>2015-06-04 14:42:00 -0700</start_time>
   <end_time>2015-06-04 15:48:00 -0700</end_time>
   <start_over>false</start_over>
   </filter_settings>
  </filter>
```

Topics

- The Live to VOD filter Element
- The Live to VOD filter_settingsElement

The Live to VOD `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only.

Tag	Type	Description
		The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. This customized address including the `output_url` is automatically create and stored in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.

Tag	Type	Description
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The Live to VOD `filter_settings`Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
`allow_url_start_end_param`	boolean	This tag controls a feature that allows the start and end times to be taken from the content request URL rather than being hard-coded into

Tag	Type	Description
		the filter via the `end_time` and `start_time` tags. • **`true`**: Enable this feature. In this case, any values in `end_time` and `start_time` are ignored. • **`false`**: Disable this feature. See Implementing URL Start/End Times for more information.
`start_time`	Datetime	The calendar date and time that the live-to-VOD clip starts. See `end_time` for the rules.

Tag	Type	Description
`end_time`	Datetime	The date and time that the live-to-VOD clip ends, formatted according to the `.iso` 8601 standard: `yyyy-mm-dd <space> hh:mm:ss <space> <offset>` • Where <offset> is the offset for the desired time zone. • If no offset is entered, the offset for the time zone specified in the Delta settings is assumed. The time you enter here is converted to UTC and then converted back to the `.iso` 8601 standard, with the offset always set to the time zone specified in Delta. If, for example, you enter a different time zone when you create the filter, when you GET the filter, the time is adjusted. If you are in the -0700 time zone but you enter “20:00 -0500”. If you then send a GET request, the time is shown as “18:00 -0700”.

Tag	Type	Description
`start_over`	boolean	Read-only. When `manifest_scope` is set to **`start_over`**, this tag is true. Otherwise, it's false.
`manifest_scope`	string	This tag controls playback behavior during the live-to-VOD window. This tag has no effect after the end time of the window. • **`default`**: Playback starts at the current time. • **`start_over`**: Playback starts at the beginning of the live-to-VOD window. • **`index_duration`**: Playback starts at the live point. The manifest is fixed length equal to the package filter's index duration.
`frame_accurate`	boolean	This tag indicates enables you to specify the frame within the indicated start and end seconds when the VOD clip will start and end. Use **`true`** to enable frame-accurate live-to-VOD. Not available with URL start/end parameters.

Tag	Type	Description
`start_frame`	integer	Indicates the frame within the specified start second that the VOD clip should start. This setting applies only when `frame_accurate` is **`true`**.
`end_frame`	integer	Indicates the frame within the specified end second that the VOD clip should start. This setting applies only when `frame_accurate` is **`true`**.
`lowest_framerate_numerator`	integer	The numerator for the lowest framerate. For example, 30 frames per second is a numerator of 30 and a denominator of 1. 20.334 frames per second is a numerator of 20334 and a denominator of 1000.
`lowest_framerate_denominator`	integer	The denominator for the lowest framerate. For example, 30 frames per second is a numerator of 30 and a denominator of 1. 20.334 frames per second is a numerator of 20334 and a denominator of 1000.

Implementing URL Start/End Times

Playback devices can include URL start and end parameters to specify the window of content for playback. This section describes how you can implement these parameters.

> **Warning**
>
> The filter playlist has to be regenerated (which can cause some players to stop playing content) if:
>
> - The `start_time` is changed.
> - The current `end_time` is in the past and is updated.
> - The new end_time is in the past.

When the `allow_url_start_end_param` tag is disabled, you must set up your content-requesting system to compose the content request URL to match the format of the endpoint tag or `output_url` tag (as described in the section called "The Live to VOD `filter` Element").

- For `endpoint`:

  ```
  http://Delta IP address:8080/out/i/filter ID.<extension>
  ```

  Example:

  ```
  http://10.24.34.2/out/i/39.mpd
  ```

- For `output_url`:

  ```
  http://Delta IP address:8080/out/u/<output_url>.<extension>
  ```

  Example:

  ```
  http://10.24.34.2/out/u/livecurling.mpd
  ```

When you enable URL start and end parameters, you must set up your content-requesting system to include the start and end time in the content request URL.

You can specify the URL using standard parameter notation:

```
<default or custom URL>?start=start time&end=end time
```

Or you can specify the URL with parameters as path elements:

```
<first part of default or custom URL>/start/start time/end/end time/remainder of URL;
```

With both notations, the times can be specified as:

- ISO 8601 dates. Example: 2015-04-27T17:30:00+00:00
- POSIX time. Example: 1430155800

The following is an example of parameter notation when your content-requesting system uses the endpoint format to access content (and assuming that the filter ID is 239):

```
http://10.24.34.2/out/i/239.mpd?
start=2015-04-27T17:30:00+00:00&end=2015-04-27T18:00:00+00:00
```

The following is an example of path elements notation when your content-requesting system uses the output_url format to access content (and assuming the output_url value is *livecurling*):

```
http://10.24.34.2/out/start/1430155800/end/1430157600/u/livecurling.mpd
```

PUT: Create a Live to VOD Catalog Output Filter

The Live to VOD Catalog output filter creates a live-to-VOD (video on demand) catalog asset from a portion of a live stream. To create the output filter, send a PUT request for the content object that corresponds to the live asset. The following sections describe how to format the request.

For information about VOD catalog, see Working with VOD Catalog Assets in AWS Elemental Delta.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID/filters
```

Body of HTTP

The body of your request is XML content consisting of one `filters` container that holds one `filter` element with several tags and the following:

- One `filter_settings` element that contains one tag. See the section called "The Live to VOD Catalog `filter_settings` Element".

Request Example

See the Live to VOD Request Example for a general processing output filter request.

Response

The response repeats back the filter that you created, with the addition of the following tags:

- `id`: Unique IDs for the filter and each sub-element as described in this section.
- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI, as follows:

```
http://<Delta IP address>:8080/out/i/<filter ID>.<extension>
```

  where `filter ID` is the ID for this filter, assigned once the filter is created.

- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if `output_uri` contains a value. Contains the custom endpoint URI, as follows:

```
http://<Delta IP address:8080>/out/u/<output_url>.<extension>
```

This example response is a representation. The `default_endpoint` now shows a value. The `filter_settings` shows the default values. The `filter`, `filter_settings`, `stream_set`, `video_track`, and `audio_track` elements are all assigned unique IDs. Your results may vary.

Topics

- The Live to VOD Catalog filter Element
- The Live to VOD Catalog filter_settings Element

The Live to VOD Catalog `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only.

Tag	Type	Description
		The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. This customized address including the `output_url` is automatically create and stored in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.

Tag	Type	Description
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The Live to VOD Catalog `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
`start_time`	Datetime	The calendar date and time that Delta promotes the asset to a VOD catalog asset. See `end_time` for the rules.

Tag	Type	Description
`end_time`	Datetime	The date and time that the live-to-VOD clip ends, formatted according to the `.iso` 8601 standard: `yyyy-mm-dd <space> hh:mm:ss <space> <offset>` • Where <offset> is the offset for the desired time zone. • If no offset is entered, the offset for the time zone specified in the Delta settings is assumed. The time you enter here is converted to UTC and then converted back to the `.iso` 8601 standard, with the offset always set to the time zone specified in Delta. If, for example, you enter a different time zone when you create the filter, when you GET the filter, the time is adjusted. If you are in the -0700 time zone but you enter “20:00 -0500”. If you then send a GET request, the time is shown as “18:00 -0700”.

Tag	Type	Description
`expiration_time`	Datetime	Calendar date/time that the asset will be promoted to a VOD Catalog asset. See `end_time` for the rules. The Live to VOD Catalog filter is removed upon expiration.
`promote_on_expiration`	boolean	This tag indicates what Delta does with the live content when it expires. • **`true`** is the default and means that the specified live-to-VOD window will be promoted to VOD catalog upon expiration. • **`false`** means that the content within the specified window is removed.
`name`	string	A unique name for the VOD Catalog content. When using replacement tokens, the `name` value replaces all instances of $name$ on output filters at egress.

Tag	Type	Description
`content_alias`	string	An optional string you can define to take place of the `vod_catalog_content_path` in the playback URL. When you use a `content_alias`, the URL is takes the following form: `http://Delta IP address:8080/out/valias/<defined content_alias>/i/filter ID.<extension>` The string must be unique, or promotion to VOD catalog content fails.
`resource_id`	string	The `resource_id` value replaces all instances of $resourceid$ on encryption output filters at egress. This allows you flexibility in the output template so that you can provide a unique value for DRM and encryption settings on each endpoint. For more information, see *Output Template Replacement Tokens* in the AWS Elemental Delta 2.3 User Guide.

Tag	Type	Description
`resource_id_alt`	string	The `resource_id_alt` value replaces all instances of $resourceid_alt$ on encryption output filters at egress. It works the same way as `resource_id` (above) and acts as an additional unique value on encryption output filters. For more information, see *Output Template Replacement Tokens* in the AWS Elemental Delta 2.3 User Guide.

PUT: Create a Preroll Ad Insertion or Postroll Ad Insertion Output Filter

The Preroll Ad Insertion and Postroll Ad insertion output filters insert ads before the content starts playing (pre-roll) or after it has finished playing (post-roll). To create the output filter, send a PUT request for the content object that corresponds to the content where the ads are inserted. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID/filters
```

Body of HTTP

The body of your request is XML content consisting of one `filters` container that holds one `filter` element with several tags and the following:

- One `filter_settings` element that contains one tag. See the section called "The Preroll Ad Insertion and Postroll Ad Insertion `filter_settings` Element".

Request Example

See the Live to VOD Request Example for a general processing output filter request.

Response

The response repeats back the filter that you created, with the addition of the following tags:

- `id`: Unique IDs for the filter and each sub-element as described in this section.
- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI, as follows:

  ```
  http://<Delta IP address>:8080/out/i/<filter ID>.<extension>
  ```

  where `filter ID` is the ID for this filter, assigned once the filter is created.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if `output_uri` contains a value. Contains the custom endpoint URI, as follows:

  ```
  http://<Delta IP address:8080>/out/u/<output_url>.<extension>
  ```

This example response is a representation. The `default_endpoint` now shows a value. The `filter_settings` shows the default values. The `filter`, `filter_settings`, `stream_set`, `video_track`, and `audio_track` elements are all assigned unique IDs. Your results may vary.

Topics

- The Preroll Ad Insertion and Postroll Ad Insertion filter Element
- The Preroll Ad Insertion and Postroll Ad Insertion filter_settings Element

The Preroll Ad Insertion and Postroll Ad Insertion `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only.

Tag	Type	Description
		Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only. The Delta-assigned default playback endpoint for the output filter.

Tag	Type	Description
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. This customized address including the `output_url` is automatically create and stored in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.

Tag	Type	Description
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The Preroll Ad Insertion and Postroll Ad Insertion `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
`ad_source_type`	string	This tag identifies the source of the ad content.

Tag	Type	Description
		• **`static`**: The ad content is Delta content. • **`vast`**: The ad content is being provided by a VAST server.
`ad_source_settings`	object	This object holds tags that describe the source of the ads. For available tags, see the following table.

The `ad_source_settings` Element

The `ad_source_settings` tag describes where you store the ads that Delta uses for preroll and postroll ad insertion. The following table describes the tags available for `ad_source_settings`.

Tag	Type	Description
`ad_sources`	string	This tag provides informati on about the source of the ads, based on the value of `ad_source_type` . • If `ad_source_type` is **`static`**: Enter a comma-separated list of the IDs (from the `id` tag) or aliases (from the `alias` tag) of the Delta content object to use as ad content. • If `ad_source_type` is **`vast`**: Enter a list of ad content tags (for static content) that Delta can use

Tag	Type	Description
		if it can't reach the VAST server. Advertising content should be VOD content-formatted to match ad avail timing. For example, if ad avails are 60 seconds, ad content should be in 15-, 30-, or 60-second intervals. Delta moves through the list of content as ad avails appear in the stream. When all specified sources have been inserted into the stream once, Delta restarts at the beginning of the list.
`campaign_id`	integer	If applicable, the ad campaign ID that you use with the VAST server. This ID is valid only if `ad_source_type` is **`vast`**, in which case it is optional.
`server`	location	The location of the VAST server. This value is required only if `ad_source_type` is **`vast`**. For contents of the location type, see Location Type Elements.

PUT: Create a Time Delay Output Filter

The Time Delay output filter delays the availability of packaged content. To create the output filter, send a PUT request for the content object that corresponds to the content that you're delaying. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID/filters
```

Body of HTTP

The body of your request is XML content consisting of one `filters` container that holds one `filter` element with several tags and the following:

- One `filter_settings` element that contains one tag. See the section called "The Time Delay `filter_settings` Element".

Request Example

See the Live to VOD Request Example for a general processing output filter request.

Response

The response repeats back the filter that you created, with the addition of the following tags:

- `id`: Unique IDs for the filter and each sub-element as described in this section.
- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI, as follows:

  ```
  http://<Delta IP address>:8080/out/i/<filter ID>.<extension>
  ```

  where `filter ID` is the ID for this filter, assigned once the filter is created.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if `output_uri` contains a value. Contains the custom endpoint URI, as follows:

  ```
  http://<Delta IP address:8080>/out/u/<output_url>.<extension>
  ```

This example response is a representation. The `default_endpoint` now shows a value. The `filter_settings` shows the default values. The `filter`, `filter_settings`, `stream_set`, `video_track`, and `audio_track` elements are all assigned unique IDs. Your results may vary.

Topics

- The Time Delay filter Element
- The Time Delay filter_settings Element

The Time Delay `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only. Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.

Tag	Type	Description
`default_endpoint_uri`	string	Read-only. The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. This customized address including the `output_url` is automatically create and stored in the `custom_endpoint_uri` tag.
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.

Tag	Type	Description
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).

Tag	Type	Description
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The Time Delay `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Note that when the `time_source` on the input filter is set to **`embedded`**, the time must be in UTC for the Time Delay output filter to work correctly.

Tag	Type	Description
`delay_seconds`	integer	The number of seconds to that the playback device should buffer the content before playing it back.

PUT: Create a VOD Clip Output Filter

The VOD Clip output filter creates a smaller video on demand (VOD) asset from a portion of an existing VOD asset. To create the output filter, send a PUT request for the content object that corresponds to the larger VOD asset. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID/filters
```

Body of HTTP

The body of your request is XML content consisting of one `filters` container that holds one `filter` element with several tags and the following:

- One `filter_settings` element that contains one tag. See the section called "The VOD Clip `filter_settings` Element".

Request Example

See the Live to VOD Request Example for a general processing output filter request.

Response

The response repeats back the filter that you created, with the addition of the following tags:

- `id`: Unique IDs for the filter and each sub-element as described in this section.
- `default_endpoint_uri`: Included if `endpoint` is **`true`**. Contains the default endpoint URI, as follows:

  ```
  http://<Delta IP address>:8080/out/i/<filter ID>.<extension>
  ```

  where `filter ID` is the ID for this filter, assigned once the filter is created.
- `custom_endpoint_uri`: Included if `endpoint` is **`true`** and if `output_uri` contains a value. Contains the custom endpoint URI, as follows:

  ```
  http://<Delta IP address:8080>/out/u/<output_url>.<extension>
  ```

This example response is a representation. The `default_endpoint` now shows a value. The `filter_settings` shows the default values. The `filter`, `filter_settings`, `stream_set`, `video_track`, and `audio_track` elements are all assigned unique IDs. Your results may vary.

Topics

- The VOD Clip filter Element
- The VOD Clip filter_settings Element

The VOD Clip `filter` Element

All output filters contain the tags listed in the following table. They are shown for each output filter section for easy reference. The tags are sorted in the order that they typically appear in a GET response.

Tag	Type	Description
`id`	integer	Read-only. Delta-assigned numeric value for the output filter.
`label`	string	Read-only.

Tag	Type	Description
		Delta-assigned filter name for the output filter.
`parent_id`	integer	To create this filter as a top-level filter, omit this tag. To attach this fiilter after another filter, specify the ID of that filter.
`endpoint`	boolean	**`true`** means that this filter has an endpoint. A player can access the content at this point in the output filter tree. **`false`** means that the filter does not have an endpoint. The player access content from a filter later in the tree.
`default_endpoint_uri`	string	Read-only. The Delta-assigned default playback endpoint for the output filter.
`output_url`	string	If `endpoint` is **`true`**, provide additional context to the endpoint to customize it. This customized address including the `output_url` is automatically create and stored in the `custom_endpoint_uri` tag.

Tag	Type	Description
`custom_endpoint_uri`	string	Read-only. The custom playback endpoint for the output filter, including the value you use for `output_url` .
`aliased_default_endpoint_uri`	string	Read-only. VOD Catalog content only. The Delta-assigned default playback endpoint for the output filter, including the VOD Catalog content alias.
`aliased_custom_endpoint_uri`	integer	Read-only. VOD Catalog content only. The custom playback endpoint for the output filter, including the value you use for `output_url` and the VOD Catalog content alias.
`ancestry`	string	Read-only. The IDs of the filters that precede this one in the output filter branch. A `nil` value indicates that this is a top-level (parent) filter.

Tag	Type	Description
`url_extension`	string	Read-only. Extension of the ingested content (such as .m3u8 or .mpd).
`description`	string	Any descriptive information that you want to add for this output filter. This informati on is useful for creating cross-references to data in other systems, such as a content management system (CMS). The `description` also helps to distinguish between multiple filters of the same type, each with different output characteristics.

Tag	Type	Description
`use_default_stream_sets`	boolean	**`true`** indicates that the upstream system (such as AWS Elemental Live) is responsible for creating the final desired stream sets. Delta automatically parses the video and audio streams in the input and creates corresponding stream sets in the output. Enter `false` to manually create the stream sets if there are subtitle tracks in the input, or if you want to create your own stream sets that are different from the input. For information about creating stream sets, see The `stream_set` Element.

The VOD Clip `filter_settings` Element

The `filter_settings` element is required. Even if you want to use the default value for all tags, you must include a `filter_settings` element in the XML body. However, it can be empty. The following table describes the tags that are available for the `filter_settings` element.

Tag	Type	Description
`allow_url_start_end_param`	boolean	This tag controls a feature that allows the start and end times to be taken from the content request URL rather than being hard-coded into

Tag	Type	Description
		the filter via the `end_time` and `start_time` tags. • **`true`**: Enable this feature. In this case, any values in `end_time` and `start_time` are ignored. • **`false`**: Disable this feature. See Implementing URL Start/End Times for more information.
`start_time`	Datetime	The calendar date and time that the live-to-VOD clip starts. See `end_time` for the rules.

Tag	Type	Description
`end_time`	Datetime	The date and time that the live-to-VOD clip ends, formatted according to the `.iso` 8601 standard: `yyyy-mm-dd <space> hh:mm:ss <space> <offset>` • Where <offset> is the offset for the desired time zone. • If no offset is entered, the offset for the time zone specified in the Delta settings is assumed. The time you enter here is converted to UTC and then converted back to the `.iso` 8601 standard, with the offset always set to the time zone specified in Delta. If, for example, you enter a different time zone when you create the filter, when you GET the filter, the time is adjusted. If you are in the -0700 time zone but you enter “20:00 -0500”. If you then send a GET request, the time is shown as “18:00 -0700”.

Implementing URL Start/End Times

Playback devices can include URL start and end parameters to specify the window of content for playback. This section describes how you can implement these parameters.

> **Warning**
>
> The filter playlist has to be regenerated (which can cause some players to stop playing content) if:
>
> - The `start_time` is changed.
> - The current `end_time` is in the past and is updated.
> - The new end_time is in the past.

When the `allow_url_start_end_param` tag is disabled, you must set up your content-requesting system to compose the content request URL to match the format of the endpoint tag or `output_url` tag (as described in the section called "The Live to VOD `filter` Element").

- For `endpoint`:

  ```
  http://Delta IP address:8080/out/i/filter ID.<extension>
  ```

  Example:

  ```
  http://10.24.34.2/out/i/39.mpd
  ```

- For `output_url`:

  ```
  http://Delta IP address:8080/out/u/<output_url>.<extension>
  ```

  Example:

  ```
  http://10.24.34.2/out/u/livecurling.mpd
  ```

When you enable URL start and end parameters, you must set up your content-requesting system to include the start and end time in the content request URL.

The URL can be specified using standard parameter notation:

```
default or custom URL?start=start time&end=end time
```

Or it can be specified with parameters as path elements of the URL:

```
first part of default or custom URL/start/start time/end/end time/remainder of URL;
```

With both notations, the times can be specified as:

- ISO 8601 dates. Example: 2015-04-27T17:30:00+00:00
- POSIX time. Example: 1430155800

Example of parameter notation when your content-requesting system uses the endpoint format to access content (and assuming that the filter ID is 239):

```
http://10.24.34.2/out/i/239.mpd?
start=2015-04-27T17:30:00+00:00&end=2015-04-27T18:00:00+00:00
```

Example of path elements notation when your content-requesting system uses the `output_url` format to access content (and assuming the `output_url` value is **`livecurling`**):

```
http://10.24.34.2/out/start/1430155800/end/1430157600/u/livecurling.mpd
```

Working with Filter Sets and Add Filters in AWS Elemental Delta

This section describes filter sets and how you can send a PUT request to add multiple filters at one time.

Topics

- Filter Sets
- PUT Add_Filters

Filter Sets

Filter sets allow you to add all or part of a filter tree in one PUT command, rather than adding each filter with individual commands. When you add filter sets, the top-most filter that you add either can either be added as a *child* of an existing filter, or it can be added as the first filter in a chain. You can create parent and child relationships by giving the parent filter a name and cross-referencing that name in its child filters.

For example, here is part of the XML needed to create the parent. The value of the `name` tag is how child filters reference this filter as their parent.

```
<filter>
   <name>filterA</name>
   .
   .
   .
</filter>
```

And here is the part of the XML needed to create the child. The `parent_filter` tag references the `name` of the parent.

```
<filter>
   <parent_filter>filterA</parent_filter>
   <endpoint>true</endpoint>
   .
   .
```

```
    .
</filter>
```

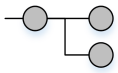
Rules

These are the placement rules for adding new output filters:

- The added filters can be linked to one location or several locations in the existing filter tree.

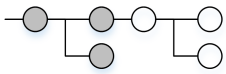
  Example Existing output filters

  This diagram shows three output filters that already exist in the filter chain.

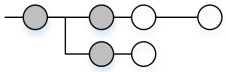
  

  Example Existing output filters with new filters added

  These diagrams show three new filters added to the filter chain.

  or

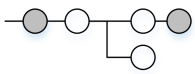
- The added filters cannot be inserted between two existing filters, but they can create a new branch.

  Example Existing output filters

  This diagram shows two output filters that already exist in the filter chain.

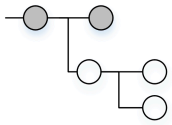
  Example Existing output filters with new filters incorrectly inserted

  This diagram shows three new output filters inserted between two existing filters. This is not a supported action.

Example Existing output filters with new filters correctly added to a branch

This diagram shows three new output filters added to a new branch. This is the supported way to add new filters.



HTTP URL

Filter trees use the normal PUT Add Output Filter command:

```
PUT http://Delta IP address:8080/contents/content ID
```

Body of HTTP

The XML consists of a tree of filters consisting of one `contents` container and one `content` element that holds:

- One `filters` container and one or more `filter` elements.

  For each *parent* filter (those output filters that are not children of another filter in the set), include the following:

  - `name`: Assign a useful name, which must be unique among the filters in this PUT. For example, **`top`**. This name does not persist.
  - `parent_id`: Specify the ID of the filter (that already exists in the content entity) to be the *parent* of this filter. You can obtain this ID using GET Output Filter List.

    Or, if this filter is a top-level output filter in the content entity, omit this tag or leave null.
  - The usual elements and tags for this output filter.

  For each *child* of a *parent* filter in the tree you are building, include the following:

  - `parent_filter`: The `name` of the *parent* filter.
  - The usual elements and tags for this output filter.

In each subsequent *child* filter in the tree you build, include the following:

- `parent_filter`: The `name` of the *child* filter that is the *parent* of this filter.
- `name`: Assign a name if this *child* filter is the *parent* of another filter. Otherwise, leave blank.

- The usual elements and tags for the output filter, as described in Creating Package Output Filters in AWS Elemental Delta, Creating Processing Output Filters in AWS Elemental Delta, and Creating Access Restriction Filters in AWS Elemental Delta.

The following table summarizes the tags that you use to create and parent and child relationships between filters.

Tag	Type	Description
`name`	string	This tag provides a temporary name for this filter, required only if this filter is the *parent* of other filters being created by this PUT command. This name does not persist. After the filter has been created, Delta assigns a unique ID to this filter.
`parent_filter`	string	When this filter is the child of another filter that you're adding with the same PUT command, this tag indicates the name of the parent filter. Enter the value of the `name` tag of the parent filter.
`parent_id`	string	When this filter is the child of another filter that already exists in the filter tree, this tag indicates the ID of the parent filter. Enter the value of the `ID` tag of the parent filter.

Tag	Type	Description
		If you're adding the parent of this filter with the same PUT command, or if this filer is to be a top-level filter, omit this tag.

Response

The response repeats back the entire filter tree for the content: both the filters that already existed and those you just created. See the following sections for request examples using filter sets.

Topics

- Example 1
- Example 2
- Example 3
- Example 4

Example 1

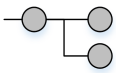
This example shows filters **A**, **B**, and **C** added. The parent of filter **A** is the filter with the ID **2**. The parent of filters **B** and **C** is filter **A**.

```
PUT http://10.24.34.2:8080/contents/9
--------------------------------
<content>
   <filters>
      <filter>
         <name>filterA</name>
         <parent_id>2</parent_id>
         <endpoint>true</endpoint>
         output_urlign</output_url>
         <filter_type>hls_package</filter_type>
         .
         .
```

```
            .
        </filter>
        <filter>
            <parent_filter>filterA</parent_filter>
            <endpoint>true</endpoint>
            output_urlign2</output_url>
            <filter_type>blackout</filter_type>
            .
            .
            .
        </filter>
        <filter>
            <parent_filter>filterA</parent_filter>
            <endpoint>true</endpoint>
            output_urlign3</output_url>
            <filter_type>remove_ad</filter_type>
            .
            .
            .
        </filter>
    </filters>
</content>
```

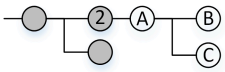
Example Before

This diagram shows three output filters that already exist in the filter chain.



Example After

This diagram shows three output filters added to the filter chain. The filter named **2** is the parent of the filter named **A**.



Example 2

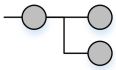
This example shows filters **A**, **B**, and **C** added. The parent of filter **A** is the filter with the ID **2**. The parent of filter **B** is filter **A**. The parent of filter **C** is filter **3**.

```
PUT http://10.24.34.2:8080/contents/9
```

```
--------------------------------
<content>
   <filters>
      <filter>
         <name>filterA</name>
         <parent_id>2</parent_id>
         <endpoint>true</endpoint>
         output_urlign</output_url>
         <filter_type>hls_package</filter_type>
         .
         .
         .
      </filter>
      <filter>
         <parent_filter>filterA</parent_filter>
         <endpoint>true</endpoint>
         output_urlign2</output_url>
         <filter_type>blackout</filter_type>
         .
         .
         .
      </filter>
      <filter>
         <parent_id>3</parent_id>
         <endpoint>false</endpoint>
         output_urlign3</output_url>
         <filter_type>remove_ad</filter_type>
         .
         .
         .
      </filter>
   </filters>
</content>
```

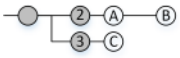
Example Before

This diagram shows three output filters that already exist in the filter chain.



Example After

This diagram shows three output filters added to the filter chain. The filter named **2** is the parent of the filter named **A**. The filter named **3** is the parent of the filter named **C**.



Example 3

This example shows filters **A**, **B**, and **C** added. Filter **A** is the inserted after the filter with the ID **1**. The parent of filter **B** is filter **A**. Filter **C** is inserted after filter **A**.

```
PUT http://10.24.34.2:8080/contents/9
--------------------------------
<content>
   <filters>
      <filter>
         <name>filterA</name>
         <parent_id>1</parent_id>
         <endpoint>true</endpoint>
         output_urlign</output_url>
         <filter_type>hls_package</filter_type>
         .
         .
         .
      </filter>
      <filter>
         <parent_filter>filterA</parent_filter>
         <endpoint>true</endpoint>
         output_urlign2</output_url>
         <filter_type>blackout</filter_type>
         .
         .
         .
      </filter>
      <filter>
         <parent_filter>filterA</parent_filter>
         <endpoint>true</endpoint>
         output_urlign3</output_url>
         <filter_type>remove_ad</filter_type>
         .
         .
         .
```

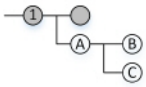
```
      </filter>
   </filters>
</content>
```

Example Before

This diagram shows two output filters that already exist in the filter chain.

Example After

This diagram shows three output filters added to the filter chain. The filter named **A** is inserted after filter **1**. The filter named **A** is the parent of the filter named **B**.



Example 4

This example shows filters **A**, **B**, and **C** added. Filter **A** is a top-level filter and is the parent of filters **B** and **C**.

```
PUT http://10.24.34.2:8080/contents/9
--------------------------------
<content>
   <filters>
      <filter>
         <name>filterA</name>
         <parent_id/>
         <endpoint>true</endpoint>
         output_urlign</output_url>
         <filter_type>hls_package</filter_type>
         .
         .
         .
      </filter>
      <filter>
         <parent_filter>filterA</parent_filter>
         <endpoint>true</endpoint>
         output_urlign2</output_url>
         <filter_type>blackout</filter_type>
         .
```

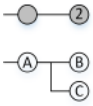
```
         .
         .
      </filter>
      <filter>
         <parent_filter>filterA</parent_filter>
         <endpoint>true</endpoint>
         output_urlign3</output_url>
         <filter_type>remove_ad</filter_type>
         .
         .
         .
      </filter>
   </filters>
</content>
```

Example Before

This diagram shows two output filters that already exist in the filter chain.

Example After

This diagram shows three output filters added to the filter chain. The filter named **A** is a top-level filter. Filters **B** and **C** are children of filter **A**.



PUT Add_Filters

The PUT Add Filters command works identically to filter sets, except that the response returns only the filters you added, not the entire filter tree. This short response is when a filter tree is very big.

HTTP URL

```
PUT http://Delta IP address:8080/contents/content ID/add_filters
```

Body of HTTP

The body of your request is XML content that is identical to the XML for filter sets. For additional information, see Filter Sets Body of HTTP.

Response

The response repeats back *only* the filters created by this PUT Add Filters command. It does not include other filters in the tree.

Working with Output Templates in AWS Elemental Delta

Output templates are considered an advanced feature. An output template is a set of output filters that you can apply to a new contents entity.

This section assumes that you are familiar with output filters.

Requirements for Output Templates

When you add an output filter to an output template, you must follow the rules for that filter, as described in Creating Access Restriction Filters in AWS Elemental Delta, Creating Package Output Filters in AWS Elemental Delta, and Creating Processing Output Filters in AWS Elemental Delta. For example, if you create an HLS output filter, the `stream_set` must contain one and only one `video_track` element.

Topics

- POST Output Template
- PUT Output Template
- GET Output Templates List
- GET an Output Template
- DELETE an Output Template

POST Output Template

To create an output template that automatically applies output filters to a contents entity, send a POST request. The following sections describe how to format the request.

HTTP URL

```
POST http://Delta IP address:8080/output_templates
```

HTTP URL for VOD Catalog

```
POST http://Delta IP address:8080/vod_output_templates
```

Body of HTTP Request

The body of your request is XML content consisting of one `output_template` or `vod_output_template` element that holds the following tags:

- `name`, which is an identifier for the output template.
- `alias`, which is an alternate identifier for the output template.

  This is the identifier that you use to reference VOD Catalog output filters.
- One or more `filter` elements (one for each desired output filter) that holds:
  - The usual tags for the filter, as described in Creating Access Restriction Filters in AWS Elemental Delta, Creating Package Output Filters in AWS Elemental Delta, and Creating Processing Output Filters in AWS Elemental Delta.
  - One `filter_settings` element that contains the usual tags for the output filter type.
  - Zero or more of the following elements (depending on the output filter type):
    - `stream_set`
    - `keyprovider_settings`
    - `nonce_key_pairs` and `nonce_key_pair`

Response

The response repeats back the data you created with the addition of the `id` tag, which is a Delta-assigned unique identifier for the template, filters, and sub-elements.

Topics

- The filter Element
- The filter_settings Element
- Stream Sets and Track Selectors

The filter Element

When you add output filters to an output template, add the typical tags for the output filter, as described in Creating Access Restriction Filters in AWS Elemental Delta, Creating Package Output Filters in AWS Elemental Delta, and Creating Processing Output Filters in AWS Elemental Delta. Additionally, always assign a name to each output filter. You can create filters individually, or as part of a filter set (recommended). For more information, see the section called “Filter Sets”.

The filter_settings Element

If the filter includes a `filter_settings` element, add the typical tags for the element.

Stream Sets and Track Selectors

If the filter is a package output filter, then you need to provide information about the stream sets to use. Specify this information in one of the following ways.

Topics

- Specify Individual Tracks
- Use the Default Stream Sets
- Use a track_selector
- The track_selector Element
- Other Elements

Specify Individual Tracks

You can specify the attributes of a specific track that exists in the content.

Typically, you can't create the stream sets this way when you create an output template because you do not yet have enough information about the tracks. But if you do want to use this method, specify the `stream_sets` in the output template in the same way as you specify them in a regular output filter.

```
<output_template>
  ...
  <filters>
    <filter>
      ...
      <filter_settings>
        ...
        <use_default_stream_sets>false</use_default_stream_sets>
        ...
      </filter_settings>
      <stream_sets>
        <stream_set>
          <id>
          .
          .
```

```
          .
          <video_tracks>
            <video_track>
              <id>
              .
              .
              .
            </video_track>
          </video_tracks>
          <audio_tracks>
            <audio_track>
              <id>
              .
              .
              .
            </audio_track>
          </audio_tracks>
          <subtitle_tracks>
            <subtitle_track>
              <id>
              .
              .
              .
            </subtitle_track>
          </subtitle_tracks>
        </stream_set>
      </stream_sets>
    </filter>
  </filters>
 </output_template>
```

Use the Default Stream Sets

If an upstream system such as AWS Elemental Live is responsible for creating the final desired stream sets, then you can set up the filter to pass through those stream sets.

To use default stream sets:

- Set the `use_default_stream_sets` tag to **`true`**.
- Omit the `stream_sets` sub-element in the output template.

```
<output_template>
```

```
  ...
  <filters>
    <filter>
      ...
      <filter_settings>
        ...
        <use_default_stream_sets>true</use_default_stream_sets>
        ...
      </filter_settings>
    </filter>
  </filters>
</output_template>
```

Use a track_selector

You can set up any track in a `stream_set` to include a `track_selector` instead of a `track_id`. For example, instead of setting up the video track include the track that has the ID 3, you can set it up to include a track that meets a specific criterion, such as the H.264 codec.

When the output template is used to create an output filter attached to content, an `xx_track` element is created if the criterion specified in that `xx_track` is met. If the criterion is not met, no `xx_track` is created.

Set up in the output template as follows:

1. Set up the `stream_set` in the usual way.
2. Set up each `xx_track` element is the same as usual, except:
   - Omit the `track_id`.
   - Include the `track_selector` element. This element is an array that specifies the selection criterion. See below.

     For HLS output filters, you can set up a rendition group using a `track_selector`. Make sure the `stream_set` includes a `rendition_group_name` and that each `audio_track` or `subtitle_track` includes a `rendition_group_value`.

     Additionally, for HLS output filters, you can set up a `stream_set` that includes an I-frame track, just like a `stream_set` created without `track_selector`. Simply set the `iframe_only` tag to **`true`**.

Example of a `track_selector`

In this example, two `audio_track` elements are specified for rendition groups. The first track includes a `track_selector` that specifies selecting any track that uses English language (eng) and the second track specifies selecting any track that uses French language (fra).

```
<output_template>
...
  <filters>
    <filter>
    ...
      <filter_settings>
        ...
        <use_default_stream_sets>false</use_default_stream_sets>
        ...
      </filter_settings>
      <stream_sets>
        <stream_set>
          <stream_index>0</stream_index>
          <rendition_group_name>audioA</rendition_group_name>
          ...
          <video_tracks/>
          <audio_tracks>
            <audio_track>
              <track_index>0</track_index>
              <rendition_group_value>engrendgroup</rendition_group_value>
              <track_selector>
                <language>eng</language>
              </track_selector>
            </audio_track>
            <audio_track>
              <track_index>1</track_index>
              <rendition_group_value>frarendgroup</rendition_group_value>
              <track_selector>
                <language>fra</language>
              </track_selector>
            </audio_track>
          </audio_tracks>
          <subtitle_tracks/>
        </stream_set>
      </stream_sets>
    </filter>
  </filters>
  ...
</output_tempaltes>
```

The track_selector Element

The `track_selector` must contain only one tag (except for `minimum/maximum_bitrate`, which is always a pair) from the following tags. In other words, it is possible to select tracks in a given `xx_track` element based on one criterion only.

Name	Type	Description
`codec`	string	Select the one track that matches the specified codec. This selector works only in ingested content that contains one matching track. If the ingested content contains two tracks with the specified codec, then Delta automatically selects the first one. • For video: H264, H265 • For audio: AACL, AC3, EAC3 • For subtitles: DFXP, TTML, WebVTT
`input_stream_index`	integer	In the incoming content, select one track, by index relative to other streams. For example, a value of 2 selects the track that is ranked second. So if the indexes are 189, 190, 191, a value of 2 selects the track with index 190.
`language`	string	Select all the tracks that match the single specified

Name	Type	Description
		language (`.iso` 639-2 three-letter code).
`maximum_bitrate`	integer	Select all tracks with a bitrate less than this bitrate. Must be used with `minimum_bitrate` to supply a range. Specify megabits as a 7-digit number or as a whole number with an **`m`**. (For example, 5 mB can be entered as 5000000 or 5m.) Specify kilobits as a 5-digit number or as a decimal with an **`m`**. (For example, with 500 kB can be entered as 500000 or 0.5m.)
`minimum_bitrate`	integer	Select all tracks with a bitrate greater than this bitrate. Must be used with `maximum_bitrate` to supply a range. Enter megabits and kilobits as specified above.
`pid`	integer	Select one track that matches the specified PID. Valid only for filters attached to HLS and TS content.

Other Elements

Create all the `keyprovider_settings` (in a DRM output filter) and `nonce_key_pairs` and `nonce_key_pair` tags (in an Authentication output filter) in the usual way described in Creating Access Restriction Filters in AWS Elemental Delta.

Example

The filters in this output template are created as a filter set Filter Sets. The top-level filter (filterA) does not have either a `parent_id` or a `parent_filter`, which indicates it is a top-level filter.

FilterA is an HLS output filter. It specifies its stream sets using track selectors, therefore, the `use_default_stream_sets` (a few lines from the top) must be false.

The second and third filters are straightforward processing filters for the HLS content. There are no special rules for creating these filters.

```
GET http://10.24.34.2:8080/output_templates
---------------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
  <filters>
    <filter>
      <name>filterA</name>
      <parent_id/>
      <endpoint>true</endpoint>
      <output_url/>
      <filter_type>hls_package</filter_type>
      <use_default_stream_sets>false</use_default_stream_sets>
      <filter_settings>
        ...
      </filter_settings>
      <stream_sets>
        <stream_set>
          <stream_index>1</stream_index>
          <video_tracks>
            <video_track>
              <track_index>0</track_index>
              <track_selector>
                <minimum_bitrate>5m</minimum_bitrate>
                <maximum_bitrate>6m</maximum_bitrate>
              </track_selector>
            </video_track>
          </video_tracks>
```

```
          <audio_tracks>
            <audio_track>
              <track_index>
              ...
            </audio_track>
          </audio_tracks>
          <subtitle_tracks>
            <subtitle_track>
              <track_index>
               ...
            </subtitle_track>
          </subtitle_tracks>
        </stream_set>
      </stream_sets>
    </filter>
    <filter>
      <parent_filter>filterA</parent_filter>
      <endpoint>true</endpoint>
      <output_url/>
      <filter_type>remove_ad</filter_type>
      ...
    </filter>
  </filters>
```

PUT Output Template

There is no PUT for output templates. If you want to modify an existing template, create a new one (perhaps by using GET with the clean attribute), then delete the unwanted template.

GET Output Templates List

Retrieve a list of output templates that are on the Delta node, as shown in the following requests.

HTTP URL

```
GET http://Delta IP address:8080/output_templates
```

HTTP URL for VOD Catalog

```
GET http://Delta IP address:8080/vod_output_templates
```

GET an Output Template

View the details of an output template, as shown in the following requests.

HTTP URL

```
GET http://Delta IP address:8080/output_templates/ID of template
```

HTTP URL for VOD Catalog

```
GET http://Delta IP address:8080/vod_output_templates/template alias
```

DELETE an Output Template

Remove an output template from the Delta node, as shown in the following requests.

HTTP URL

```
DELETE http://Delta IP address:8080/output_templates/ID of template
```

HTTP URL for VOD Catalog

```
DELETE http://Delta IP address:8080/vod_output_templates/template alias
```

Response

A 200 OK response indicates the delete was successful.

Monitoring AWS Elemental Delta

Alerts, messages, and Healthz are tools that help you monitor the health and status of Delta.

GET Alerts

To receive a list of active alerts on the node, send a GET request for the node. The alerts API provides information about current alert conditions on the Delta node. The following sections describe how to format the request.

HTTP URL

```
GET http://Delta IP address:8080/alerts
```

You can use the parameters in the following table to filter the alerts that are included in the response to the GET command.

Filter Parameter	Value	Description
`type`	string	**`all`** to display all responses . The value is case-sensitive. Use **`all`**, not **`All`**. Omit the type filter to return only active alerts.
`page`	integer	See below. Default is 1.
`per_page`	integer	See below. Default is 20.

page and per_page

You can use `page` and `per_page` optional filter parameters to narrow the results that Delta includes in the response to your GET request.

- The `per_page` parameter chunks results into groups or *pages* of a given count, with the newest entries on page 1. For example, **`30`** means chunk entries into one page for IDs 1-30, another for 31-60, and so on.

- The `page` parameter indicates that you want results for the specified page (default is page 1). For example, if `per_page` is **30**, then page 3 contains the entries that appear in positions 61 to 90. If `page` is **3**, then you will see entries that are in positions 61 to 90.

You can use either one or both of the filter parameters in your GET request.

The key use for the filter parameters is to retrieve a large number of alerts by specifying a high number in `per_page`.

Request Example

```
GET http//10.24.34.2:8080/alerts?type=all&per_page=300
```

Response

The response is XML consisting of one `alerts_list` element that holds:

One or more *`type`*`_alert` elements. Each element contains:

- A unique `id` tag.
- Several other tags that apply to the alert type.

```
<?xml version="1.0" encoding="UTF-8"?>
  <alerts_list>
   <delta_content_alert>
   <id>7</id>
   <last_set>2017-04-12T12:22:47-07:00</last_set>
   <message>Content 1 is stale</message>
   <name>Stale Content Alert</name>
   <quiet>false</quiet>
   <set>true</set>
   <content href="">1</content>
   <node>delta1</node>
   </delta_content_alert>
   <mount_point_alert>
   <id>15</id>
   <last_set>2017-04-11T15:05:56-07:00</last_set>
   <message>ERROR: Mount failed! mount error(110): Connection timed out Refer to the
 mount.cifs(8) manual page (e.g. man mount.cifs) You may have entered an incorrect
 username, password or file share. Retrying in 5 minutes</message>
   <name>Mount Point /data/mnt/folders Alert</name>
   <quiet>false</quiet>
   <set>true</set>
```

```
<file_system_mount href="">8</file_system_mount>
<node>delta1</node>
</mount_point_alert>
</alerts_list>
```

GET Messages

To receive a list of messages for the node, send a GET request. The following sections describe how to format the request.

Messages provide an audit list of events on Delta.

HTTP URL

```
GET http://Delta IP address:8080/messages
```

The GET request can include any of the following filter parameters:

Filter Parameter	Value	Description
`type`	string	One of: **`Audit`**, **`Content`**, **`Error`**, **`Warning`**. The value is case-sensitive: **`Alert`** not **`alert`**."
`page`	integer	See below. Default is 1.
`per_page`	integer	See below. Default is 20.

page and per_page

You can use `page` and `per_page` optional filter parameters to narrow the results that Delta includes in the response to your GET request.

- The `per_page` parameter chunks results into groups or *pages* of a given count, with the newest entries on page 1. For example, **`30`** means chunk entries into one page for IDs 1-30, another for 31-60, and so on.

- The `page` parameter indicates that you want results for the specified page (default is page 1). For example, if `per_page` is **30**, then page 3 contains the entries that appear in positions 61 to 90. If `page` is **3**, then you will see entries that are in positions 61 to 90.

You can use either one or both of the filter parameters in your GET request.

The key use for the filter parameters is to retrieve a large number of messages by specifying a high number in `per_page`.

Request Example

```
GET http//10.24.34.2:8080/messages?type=All&per_page=300
```

Response

XML consisting of one `messages_list` element that contains:

- One or more content elements. Each element is one of the following types: `audit`, `content`, `error`, `warning`. Each element contains:
  - A unique `id` tag.
  - Several other tags that apply to the message type.

```
<?xml version="1.0" encoding="UTF-8"?>
  <messages>
   <content>
   <code>50</code>
   <message>New Content: mendisInterview.m3u8</message>
   <id>74</id>
   </content>
   <audit>
   <code>32</code>
   <message>Node delta_02 added to cluster</message>
   <id>62</id>
   <node>delta_02</node>
   </audit>
  </messages>
   <next href="http://10.24.34.2:8080/messages?page=2"/>
```

Healthz: Get Node Health

Healthz provides information about whether a specified node is healthy, based on how long the node takes to respond to the healthz request.

You can set up your load balancer to send the healthz command to each node on a periodic basis. If the current response delay is under the value you set in `healthz_replication_delay`, then Healthz response returns an HTTP 200. If the response time is over the replication delay, then the response is an HTTP error 503 (service unavailable).

If a 503 error is returned, you may decide to set up your load balancer to stop sending requests to that node. You can then investigate the problem.

HTTP URL

```
GET http://Delta IP address:8080/healthz
```

Response

- 200 OK if the delay for this node is under the cutoff.
- Error 503 (service unavailable) if the delay for this node is over the cutoff.

Configuring AWS Elemental Delta

The following sections describe how to configure a variety of settings on the Delta node.

Topics

- Querying Settings
- Managing Amazon S3 Credentials
- Configuring Mount Points
- Configuring an AWS Elemental Delta Cluster
- Working with Authorized Users

Querying Settings

The settings element holds information about parameters that configure Delta. With the REST API, you can only query settings. To set or modify them, you must use the Delta web interface.

Topics

- GET Settings
- GET General Settings
- GET SNMP Configuration

GET Settings

To retrieve the current settings for the Delta node, send a GET request. The following sections describe how to format the request.

HTTP URL

```
GET http://Delta IP address:8080/settings
```

Response

The response is XML content consisting of one `settings` element that holds:

- One `timezone` tag.
- One `network_config` element that holds several tags.

- One `firewall_config` element that holds several tags.
- Zero or one `mount_point` elements that each holds several tags.
- One `sequencer_config` element that holds several tags.

This example is a representation. Your results may vary.

```
GET http://10.24.34.2:8080/settings
-------------------------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<settings>
 <timezone>Pacific Time (US &amp; Canada)</timezone>
 <network_config>
     <hostname>delta01</hostname>
     <pending_update>false</pending_update>
     <nameserver>10.6.16.10</nameserver>
     <ntp_server>0.centos.pool.ntp.org</ntp_server>
     <ntp_server>1.centos.pool.ntp.org</ntp_server>
     <ntp_server>2.centos.pool.ntp.org</ntp_server>
     <eth_config>
         <address_mode>dhcp</address_mode>
         <description></description>
         <eth_dev>eth0</eth_dev>
         <management>true</management>
         <parent_eth_config_id nil="true"/>
         <name>eth0</name>
     </eth_config>
     <eth_config>
         <address_mode>dhcp</address_mode>
         <description></description>
         <eth_dev>eth1</eth_dev>
         <management>false</management>
         <parent_eth_config_id nil="true"/>
         <name>eth1</name>
         <route_config>
         <gateway></gateway>
         <netmask>240.0.0.0</netmask>
         <network>225.225.255.255</network>
         </route_config>
     </eth_config>
 </network_config>
 <mount_point_config>
     <file_system_mounts>
         <file_system_mount>
```

```
            <cluster_file_system_mount_id nil="true"/>
            <file_system_type>cifs</file_system_type>
            <id>8</id>
            <mount_folder>folders</mount_folder>
            <password>pword</password>
            <server_share>//server/share</server_share>
            <username>username</username>
            <status>mounting</status>
        </file_system_mount>
    </file_system_mounts>
 </mount_point_config>
 <firewall_config>
    <firewall_on>true</firewall_on>
    <firewall_ports>
        <firewall_port>
            <port>5432</port>
            <port_type>tcp</port_type>
        </firewall_port>
   </fireall_ports>
 </firewall_config>
 <sequencer_config>
    <cluster_sequencer_config_id>1</cluster_sequencer_config_id>
    <copy_local_dir>/data/local_sources/</copy_local_dir>
    <cpu_load_factor>40</cpu_load_factor>
    <disable_profiles_and_levels_messaging>true</
disable_profiles_and_levels_messaging>
    <exclude_gpu_0>false</exclude_gpu_0>
    <exclude_gpu_1>false</exclude_gpu_1>
    <exclude_gpu_2>false</exclude_gpu_2>
    <exclude_gpu_3>false</exclude_gpu_3>
    <first_msg_timeout>60</first_msg_timeout>
    <id>78</id>
    <ingest_parser_enabled>false</ingest_parser_enabled>
    <is_updated>false</is_updated>
    <job_poll_rate>1</job_poll_rate>
    <jobcancelled_poll_rate>1</jobcancelled_poll_rate>
    <kclks_per_gwu>86760</kclks_per_gwu>
    <max_jobs>4</max_jobs>
    <media_info_timeout>15</media_info_timeout>
    <node_id>78</node_id>
    <pct_rt_threshold>100</pct_rt_threshold>
    <restart_work_after_power_loss>true</restart_work_after_power_loss>
    <rtmp_input>true</rtmp_input>
    <rtmp_input_port>1935</rtmp_input_port>
```

```
    <stalejob_age>600</stalejob_age>
    <stalejob_poll_rate>5</stalejob_poll_rate>
    <suppress_deprecation_warnings>false</suppress_deprecation_warnings>
    <throttle_poll_rate>5</throttle_poll_rate>
    <updated_at>2016-11-29T16:16:12-08:00</updated_at>
    <use_cpu_rate>800000</use_cpu_rate>
    <use_cpu_saturated>false</use_cpu_saturated>
    <use_cpu_size>307200</use_cpu_size>
    <use_progressive_reader>true</use_progressive_reader>
    <watchfolder_poll_rate>10</watchfolder_poll_rate>
 </sequencer_config>
</settings>
```

GET General Settings

To retrieve the current general settings for the Delta node, send a GET request. The following sections describe how to format the request.

HTTP URL

```
GET http://Delta IP address:8080/settings/general_settings
```

Response

The response for GET General Settings is XML content consisting of one `general_settings` element that holds:

- One `timezone` tag.
- One `public_endpoint` tag.
- One `vod_catalog_metadata_storage_location` tag.
- One `vod_catalog_metadata_storage_location_credential_id` tag.
- One `cluster` element that contains several tags.

This example is a representation. Your results may vary.

```
GET http://10.24.34.2:8080/settings/general_settings
-------------------------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<general_settings>
 <timezone>Pacific Time (US &amp; Canada)</timezone>
```

```
 <public_endpoint nil="true"/>
 <vod_catalog_metadata_storage_location>/data/server/vod_egress_data/</
vod_catalog_metadata_storage_location>
 <vod_catalog_metadata_storage_location_credential_id nil="true"/>
 <cluster>
     <id>1</id>
     <database_backup_minutes>1440</database_backup_minutes>
     <database_backup_num_backups>5</database_backup_num_backups>
     <database_backup_dir>/home/elemental/database_backups</database_backup_dir>
     <global_alert_notification>
         <created_at nil="true"/>
         <email nil="true"/>
         <event/>
         <id nil="true"/>
         <notifiable_id>1</notifiable_id>
         <notifiable_type>Cluster</notifiable_type>
         <updated_at nil="true"/>
         <web_callback_url nil="true"/>
     </global_alert_notification>
 </cluster>
</general_settings>
```

GET SNMP Configuration

To retrieve the current configuration for SNMP, send a GET request. The following sections describe how to format the request.

HTTP URL

```
GET http://Delta IP address:8080/settings/snmp
```

Response

The response for GET SNMP configuration is XML content consisting of one `snmp_config` element that holds several tags.

This example is a representation. Your results may vary.

```
GET http://10.24.34.2:8080/settings/snmp
-------------------------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
```

```
<snmp_config>
 <snmp_access>true</snmp_access>
 <snmp_trap_alerts>true</snmp_trap_alerts>
 <snmp_trap_community>Public</snmp_trap_community>
 <snmp_trap_host>10.10.5.4</snmp_trap_host>
 <snmp_trap_port>162</snmp_trap_port>
```

Managing Amazon S3 Credentials

Create and maintain Amazon S3 credentials from AWS Elemental Delta when using S3 as a content repository.

Topics

- POST Amazon S3 Credentials
- PUT Amazon S3 Credentials
- GET Amazon S3 Credentials
- DELETE Amazon S3 Credentials

POST Amazon S3 Credentials

To add an AWS user's credentials to AWS Elemental Delta so Delta can use these credentials to access Amazon S3 for content storage, send a POST request. The following sections describe how to format the request.

HTTP URL

```
POST http://Delta IP address:8080/settings/aws_credentials
```

Body of HTTP

The body of your request is XML content consisting of one `aws_credential` element that holds the following tags:

Tag	Type	Description
name	string	The name for the AWS user as configured in

Tag	Type	Description
		AWS Identity and Access Management (IAM).
`access_key`	string	The access key ID for the user, downloaded from IAM.
`secret_key`	string	The secret access key for the user, downloaded from IAM.

Request Example

The following example adds credentials for `credential_user_1` .

```
POST http://10.24.34.2:8080/settings/aws_credentials
---------------------------------------------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<aws_credential
 <name>credential_user_1</name>
 <access_key>XXX00XX0X00XXX</access_key>
    <secret_key>YY111Y1YYY111YY</secret_key>
</aws_credential>
```

Response

The response repeats back the data that you posted, with the addition of a unique ID assigned by the system.

PUT Amazon S3 Credentials

To modify the attributes of the specified AWS user, send a PUT request. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/settings/aws_credentials/id
```

where *`id`* is the unique ID of the `aws_credential` to modify.

Body of HTTP

The body of your request is XML content consisting of only the tags to change. For tag details, see POST Amazon S3 Credentials.

Request Example

In this example, the request changes the access_key of the AWS user with the ID 14. The value changes to `987HK9jfhq9`.

```
PUT http://10.24.34.2:8080/aws_credential/14
-----------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<aws_credentials>
 <aws_credential>
     <access_key>987HK9jfhq9</access_key>
 </aws_credential>
</aws_credentials>
```

Response

The response repeats back all the data in the AWS credential.

GET Amazon S3 Credentials

To retrieve a list of AWS users configured on the node, send a GET request. The following sections describe how to format the request.

HTTP URL

```
GET http://Delta IP address:8080/settings/aws_credentials
```

Response

The response is XML content consisting of one `aws_credentials` container that holds zero or more `aws_credential` elements, each with:

- An HREF that specifies the unique ID of the user and the product and version installed on the node.
- The tags described in POST Amazon S3 Credentials, minus the `secret_key`.

DELETE Amazon S3 Credentials

To remove the AWS user with the specified ID, send a DELETE request. The following sections describe how to format the request.

HTTP URL

```
DELETE http://Delta IP address:8080/settings/aws_credentials/id
```

where *id* is the unique ID of the `aws_credential` to delete.

Response

A 200 OK response indicates the delete was successful.

Configuring Mount Points

Mount points are remote servers that the Delta node can access. The following sections describe how to create and manage mount points.

Topics

- POST Mount Point
- GET Mount Point List
- DELETE Mount Point

POST Mount Point

To add a mount point on the Delta node, send a POST request. The following sections describe how to format the request.

HTTP URL

```
POST http://Delta IP address:8080/settings/mount_points
```

Body of HTTP

The body of your request is XML content consisting of one `mount_point` element that holds the following tags:

Tag	Type	Description
`file_system_type`	string	The type of remote server that is mounted on the Delta node. • **`cifs`**: Windows CIF server, or Windows or Mac SMB server. • **`nfs`**: Linux server. • **`webdav`**: DavFS server. **Important** For live workflow storage, we recommend that you use NFS or CIFS file servers. AWS Elemental Delta provides DavFS as a mounting option for a limited number of use cases, such as moving ancillary files (like DRM policy files) at a low rate. DavFS is often too slow to be used as storage for live inputs.
`mount_folder`	string	The folder on the node where the remote folder

Tag	Type	Description
		is mounted. This folder is mounted under `/data/mnt` (as indicated in the path element).
`password`	string	The password used if the remote server folder requires a username and password.
`server_share`	string	The address of the folder on the remote computer that you want to make available on this node. You must format this as shown in the following request example.
`username`	string	The username used if the remote server folder requires a username and password.

Request Example

This request adds a CIFS server to the Delta node.

```
POST http://10.24.34.2:8080/settings/mount_points
-------------------------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<mount_point>
 <file_system_type>cifs</file_system_type>
 <mount_folder>folders</mount_folder>
 <password>pword</password>
 <server_share>//server/share</server_share>
 <username>username</username>
</mount_point>
```

Response

A 200 OK response is received.

GET Mount Point List

To retrieve a list of all mount points on the Delta node, send a GET request. The following sections describe how to format the request.

HTTP URL

```
GET http://Delta IP address:8080/settings/mount_points
```

Response

The response is XML content consisting of one `mount_point_config` element that holds:

- One `types` element containing the type of remote servers.
- One `path` element containing the path to the folder where the remote server is mounted (always /data/mnt).
- Zero or more `mounts` elements, each with:
  - One `mount` element, including:
    - Tags as described in POST Mount Point minus the `password`, and
    - `id` and `status` of the mount point

Note that the response to a GET request returns `mount` but `mount_point` is required when sending a POST request.

This example is a representation. Your results may vary.

```
GET http://10.24.34.2:8080/settings/mount_points
-------------------------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<mount_point_config>
 <types type="array">
     <type>cifs</type>
     <type>nfs</type>
     <type>davfs</type>
 </types>
 <path>/data/mnt/</path>
 <mounts type="array">
     <mount>
         <file_system_type>cifs</file_system_type>
         <id>2</id>
         <mount_folder>mountfolder</mount_folder>
```

```
        <password>pword</password>
        <server_share>//<server_share>/<test></server_share>
        <username>username</username>
        <status>mounting</status>
    </mount>
 </mounts>
</mount_point_config>
```

DELETE Mount Point

To remove the mount point with the specified ID, send a DELETE request. The following sections describe how to format the request.

HTTP URL

```
DELETE http://Delta IP address:8080/settings/mount_points/id
```

where *id* is the system-assigned ID of the mount point to be deleted.

Response

A 200 OK response indicates the delete was successful.

Configuring an AWS Elemental Delta Cluster

You can't create a Delta cluster via the REST API. However, once the cluster has been created via the web interface, you can use the API to perform the actions in the following sections.

Topics

- GET: Query the Cluster Configuration
- GET: Query Pave Progress
- PUT: Modify the Cluster
- POST: Create a Virtual IP Address (VIP)
- PUT: Modify a VIP
- GET VIPs List
- GET a VIP
- DELETE a VIP

GET: Query the Cluster Configuration

To retrieve the configuration attributes of the cluster, send a GET request. The following sections describe how to format the request. The following sections describe how to format the request.

HTTP URL

```
GET http://Delta IP address:8080/clusters/1
```

where 1 is the ID of the cluster; there is only every one cluster and its ID is always 1.

Response

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

- An HREF that specifies the path to this filter and the product and version installed on the node.
- Several tag, as described in the following table.

Tag	Type	Description
`id`	integer	The ID for the cluster; always **1**.
`heartbeat_interval_seconds`	integer	The frequency in seconds of the heartbeat. A heartbeat is sent between the leader and secondary nodes. The secondary node uses the presence or failure of this heartbeat to assume that the leader node is (or is not) working.
`take_action_time_seconds`	integer	The number of seconds to wait after a failover has occurred before allowing another failover. This delay ensures that, in a failover,

Tag	Type	Description
		the new leader node has had time to completely gain control of the cluster. For example, if the delay is too short when node A fails and node B becomes the leader, then the system may detect that node B has failed, when, in fact, node B is not yet completely responsive because it is still setting itself up as leader.
`drop_node_after_seconds`	integer	The number of seconds before a node is considered to have failed. If the leader node is detected as having failed (the heartbeat is not received by the secondary node), node activity fails over to the secondary node after this many seconds.
`clustermgr_interval_seconds`	integer	The interval (in seconds) for performing cluster management tasks, such as checking for node timeouts.
`snapshot_rerequest_time_seconds`	integer	The time to wait for a response to a snapshot request before requesting it again. Default is 120.

Tag	Type	Description
`snapshot_maxage_seconds`	integer	The maximum age of the request. If a snapshot request is sent to the primary node within this amount of time (age) since the last request, the response from the last request is used. If all requests exceed the maximum indicated age, then a new response is provided. Default is 3600.

Tag	Type	Description
healthz_replication_delay	integer	The current delay of replicating data between the leader node and secondary nodes. The databases on the secondary and egress nodes are continually replicating the data from the leader node. Typically, the data is replicated from the leader node in a few seconds. The current delay is displayed on the Nodes screen in the Replication Lag field for each node. A long delay may indicate a problem in the cluster. You can set the cutoff for an unacceptable delay in this field (the default is 10 seconds). You can then set up your load balancer to send a healthz REST API command, as described in Healthz: Get Node Health.

This example is a representation. Your results may vary.

```
GET http://10.24.34.2:8080/clusters/1
-----------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<cluster href="/clusters/1" product="Delta" version="2.3.0.123456">
 <id>1</id>
 <heartbeat_interval_seconds>2</heartbeat_interval_seconds>
 <take_action_time_seconds>20</take_action_time_seconds>
```

```
 <drop_node_after_seconds>10</drop_node_after_seconds>
 <clustermgr_interval_seconds>5</clustermgr_interval_seconds>
 <snapshot_rerequest_time_seconds>900</snapshot_rerequest_time_seconds>
 <snapshot_maxage_seconds>3600</snapshot_maxage_seconds>
 <healthz_replication_delay>2</healthz_replication_delay>
</cluster>
```

GET: Query Pave Progress

When the leader node fails over to the secondary node in an AWS Elemental Delta cluster, send a GET request to retrieve the pave status of the secondary node as it becomes the new leader. The following sections describe how to format the request.

HTTP URL

```
GET http://Delta IP address:8080/nodes/node_id.json
```

where *node_id* is the ID of the new leader node.

Response

The response provides information about the pave status.

```
GET http://10.24.34.2:8080/nodes/2.json
-----------------------------------
<node>
  <status>pave_in_progress</status>
  <status_text>pave progress text</status_text>
</node>
```

PUT: Modify the Cluster

To modify the configuration of an existing cluster, send a PUT request. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/clusters/1
```

where 1 is the ID of the cluster; there is only every one cluster and its ID is always 1.

Body of HTTP

The body of your request is XML content consisting of one `cluster` element that contains only the tags to change.

Request Example

```
PUT http://10.24.34.2:8080/clusters/1
-----------------------------------
<cluster>
  <heartbeat_interval_seconds>4</heartbeat_interval_seconds>
</cluster>
```

Response

The response repeats back all the data for the cluster.

This example is a representation. Your results may vary.

```
<?xml version="1.0" encoding="UTF-8"?>
<cluster href="/clusters/1" product="Delta" version="2.3.0.123456">
  <id>1</id>
  <heartbeat_interval_seconds>4</heartbeat_interval_seconds>
  <take_action_time_seconds>20</take_action_time_seconds>
  <drop_node_after_seconds>10</drop_node_after_seconds>
  <clustermgr_interval_seconds>5</clustermgr_interval_seconds>
  <snapshot_rerequest_time_seconds>900</snapshot_rerequest_time_seconds>
  <snapshot_maxage_seconds>3600</snapshot_maxage_seconds>
</cluster>
```

POST: Create a Virtual IP Address (VIP)

To create a VIP in the cluster, send a POST request. The following sections describe how to format the request.

HTTP URL

```
POST http://Delta IP address:8080/clusters/1/vips
```

where 1 is the ID of the cluster; there is only every one cluster and its ID is always 1.

Body of HTTP

The body of your request is XML content consisting of one `xml` element that contains the following tags:

Tag	Type	Description
`address`	string	An IPv4 address.
`broadcast`	string	The broadcast address for this VIP. Normally it matches the broadcast address on the interface to which the VIP is assigned .
`interface`	string	The network interface this VIP is attached to. The interface must exist and be active on all nodes in the cluster. Format as **`ethn`**. Default is eth0.
`netmask`	integer	Netmask for this VIP. Normally matches the netmask on the interface to which the VIP is assigned. Range is 0-33. Default is 24.

Request Example

This request creates a VIP with the IP address `10.4.4.4` for cluster 1.

```
POST http://10.24.34.2:8080/clusters/1/vips
-----------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<vips>
```

```
  <vip>
   <address>10.4.4.4</address>
   <netmask>255.255.255.0</netmask>
   <broadcast>10.6.6.100</broadcast>
   <interface>eth1</interface>
  </vip>
</vips>
```

Response

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

- An HREF that specifies the path to this filter and the product and version installed on the node.
- The `vip_type` tag that always specifies **`management`**.
- The data posted in the request.

PUT: Modify a VIP

To modify the attributes of an existing VIP, send a PUT request. The following sections describe how to format the request.

HTTP URL

```
PUT http://Delta IP address:8080/clusters/1/vips/vipID
```

where:

- 1 is the ID of the cluster; there is only every one cluster and its ID is always 1.
- *`vipID`* is the unique ID of the VIP to modify.

Body of HTTP

The body of your request is XML content consisting of one `vip` element that contains only the tags to change. See POST: Create a Virtual IP Address (VIP).

Request Example

This request changes the interface on the VIP to `eth0`.

```
PUT http://10.24.34.2:8080/clusters/1/vips
-----------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
```

```
<vip>
 <interface>eth0</interface>
</vip>
```

Response

The response is XML content consisting of one `vip` element that repeats back all the data for the specified VIP.

This example is a representation. Your results may vary.

```
<?xml version="1.0" encoding="UTF-8"?>
<vip href="/clusters/1/vips/1" product="Delta" version="2.3.0.123456">
 <id>1</id>
 <address>10.4.4.4</address>
 <netmask>255.255.255.0</netmask>
 <broadcast>10.6.6.100</broadcast>
 <interface>eth0</interface>
 <vip_type>management</vip_type>
</vip>
```

GET VIPs List

To retrieve a list of the VIPs in the cluster, send a GET request. The following sections describe how to format the request.

HTTP URL

```
GET http://Delta IP address:8080/clusters/1/vips
```

Where 1 is the ID of the cluster; there is only every one cluster and its ID is always 1.

Response

The response is XML content consisting of one `vips` container holding one or more `vip` elements.

This example is a representation. Your results may vary.

```
GET http://10.24.34.2:8080/clusters/1/vips
-------------------------------------------
<vips href="/clusters/1/vips" product="Delta" version="2.3.0.123456">
```

```
 <vip href="/clusters/1/vips/1" product="Delta" version="2.3.0.123456">
   <id>1</id>
   <address>10.4.4.4</address>
   <netmask>255.255.255.0</netmask>
   <broadcast>10.6.6.100</broadcast>
   <interface>eth1</interface>
   <vip_type>management</vip_type>
 </vip>
 <vip href="/clusters/1/vips/1" product="Delta" version="2.3.0.123456">
  .
  .
  .
 </vip>
</vips>
```

GET a VIP

To retrieve the attributes for one VIP, send a GET request. The following sections describe how to format the request.

HTTP URL

```
GET http://Delta IP address:8080/clusters/1/vips/vipID
```

where:

- 1 is the ID of the cluster; there is only every one cluster and its ID is always 1.
- *vipID* is the unique ID of the VIP to get.

Response

The response is XML content consisting of one `vip` element.

This example is a representation. Your results may vary.

```
GET http://10.24.34.2:8080/clusters/1/vips/1
-----------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<vip href="/clusters/1/vips/1" product="Delta" version="2.3.0.123456">
   <id>1</id>
   <address>10.4.4.4</address>
   <netmask>255.255.255.0</netmask>
```

```
    <broadcast>10.6.6.100</broadcast>
    <interface>eth1</interface>
    <vip_type>management</vip_type>
</vip>
```

DELETE a VIP

To remove the specified VIP, send a DELETE request. The following sections describe how to format the request.

HTTP URL

```
DELETE http://Delta IP address:8080/clusters/1/vips/vipID
```

where:

- 1 is the ID of the cluster; there is only every one cluster and its ID is always 1.
- *vipID* is the unique ID of the VIP to delete.

Response

A 200 OK response indicates the delete was successful.

Working with Authorized Users

Authorized users can access Delta when user authentication is enabled.

Note that if authentication is enabled, you must provide additional information in the request header. See Using the API with User Authentication Enabled.

Topics

- POST: Add an Authorized User
- GET Users
- DELETE a User

POST: Add an Authorized User

To add a user who can access Delta when user authentication is enabled, send a POST request. The following sections describe how to format the request.

HTTP URL

```
POST http://Delta IP address:8080/users
```

Body of HTTP

The body of your request is XML content consisting of one `user` element that holds the following:

- An HREF tag that specifies the path to this filter and the product and version installed on the node.
- Several tags, as described in the following table.

Tag	Type	Description
`email`	string	The email address for the user being added.
`login`	string	The username the new user uses to access Delta.
`password`	string	The password the new user uses to access Delta.
`role_id`	integer	The access level for the new user. Enter a number 1 through 4 to indicate the role. • 1 = Admin • 2 = Manager • 3 = Operator • 4 = Viewer For more information about user roles, see the AWS Elemental Delta version 2.3 Configuration Guide

Request Example

This request creates a user named `newuser`.

```
POST http://10.24.34.2:8080/users
-----------------------------------
<user>
  <email>new.user@emailaddress.com</email>
  <login>newuser</login>
  <password>newuserpassword</password>
  <role_id>4</role_id>
</user>
```

Response

The response is XML content consisting of one `user` element.

This example is a representation. Your results may vary.

```
<?xml version="1.0" encoding="UTF-8"?>
<user href="/users" product="Delta" version="2.3.0.123456">
 <id>7</id>
 <api_key>SaAoT5QUbHfRSxTa2Bwk</api_key>
 <role>Manager</role>
 <email>new.user@emailaddress.com</email>
 <login>newuser</login>
 <expires_at nil="true"/>
 <admin>false</admin>
 <password_expired>false</password_expired>
</user>
```

GET Users

To retrieve a list of authorized users who are configured on the account, send a GET request. The following sections describe how to format the request.

HTTP URL

```
GET http://Delta IP address:8080/users
```

Response

The response is XML content consisting of:

- An HREF that specifies the path to this filter and the product and version installed on the node.
- Several tags as described in the following table.

Tag	Type	Description
`email`	string	The email addressfor the new user being added.
`login`	string	The username the new user uses to access Delta.
`password`	string	The password the new user uses to access Delta.
`role_id`	integer	The access level for the new user. Enter a number 1 through 4 to indicate the role. • 1 = Admin • 2 = Manager • 3 = Operator • 4 = Viewer For more information about user roles, see the AWS Elemental Delta Configura tion Guide.

This example is a representation. Your results may vary.

```
GET http://10.24.34.2:8080/users
-------------------------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
```

```
<user href="/users" product="Delta" version="2.3.0.123456">
 <id>3</id>
 <api_key>faj90UJkljafg</api_key>
 <role>Admin</role>
 <email>user@emailaddress.com</email>
 <login>user</login>
 <expires_at nil="true"/>
 <admin>true</admin>
 <password_expired>false</password_expired>
</user>
```

DELETE a User

To remove the specified user, send a DELETE request. The following sections describe how to format the request.

HTTP URL

```
DELETE http://Delta IP address:8080/users/id
```

where *id* is the Delta-assigned ID number for the user.

Response

A 200 OK response indicates the delete was successful.

AWS Elemental Delta XML Structure

The following diagrams show the basic structure of output filters in the AWS Elemental Delta API.

Topics

- Package Output Filter
- DRM Output Filter
- Authentication Output Filters
- Processing Output Filters

Package Output Filter

This is the general XML structure of a package output filter (such as HLS).

```
<contents>
   <content>
   <id>
   .
   .
   <filters>
   <filter>
   <id>
   .
   .
   <filter_settings>
   <id>
   <stream_sets>
   <stream_set>
   <id>
   .
   .
   <video_tracks>
   <video_track>
   <id>
   .
   .
   </video_track>
   </video_tracks>
   <audio_tracks>
```

```
    <audio_track>
    <id>
    .
    .
    </audio_track>
    </audio_tracks>
    <subtitle_tracks>
    <subtitle_track>
    <id>
    .
    .
    </subtitle_track>
    </subtitle_tracks>
    </stream_set>
    </stream_sets>
    </filter>
    </filters>
    </content>
   </contents>
```

DRM Output Filter

This is the general XML structure of a digital rights management (DRM) output filter (such as Playready).

```
<contents>
    <content>
    <id>
    .
    .
    <filters>
    <filter>
    <id>
    .
    .
    <filter_settings>
    <id>
    .
    .
    <keyprovider_settings>
    <id>
    .
```

```
   .
   </keyprovider_settings>
   </filter_settings>
   </filter>
   </filters>
   </content>
  </contents>
```

Authentication Output Filters

This is the general XML structure of an authentication output filter (such as Cisco URL Signing).

```
<contents>
   <content>
   <id>
   .
   .
   <filters>
   <filter>
   <id>
   .
   .
   <filter_settings>
   <id>
   .
   .
   <nonce_key_pairs>
   <nonce_key_pair>
   <id>
   .
   .
   </nonce_key_pair>
   </nonce_key_pairs>
   <ip_ranges />
   </filter_settings>
   <stream_sets />
   </filter>
   </filters>
   </content>
  </contents>
```

Processing Output Filters

This is the general XML structure of a processing output filter (such as Live to VOD).

```
<contents>
   <content>
   <id>
   .
   .
   <filters>
   <filter>
   <id>
   .
   .
   <filter_settings>
   <id>
   .
   .
   </filter_settings>
   </filter>
   </filters>
   </content>
  </contents>
```

Document History for API Guide

The following table describes the documentation for this release of AWS Elemental Delta.

- **API version:** 2.3
- **Release notes:** AWS Elemental Delta Release Notes

The following table describes the documentation for this release of AWS Elemental Delta. For notification about updates to this documentation, you can subscribe to an RSS feed.

Change	Description	Date
Guide format conversion	This guide has been converted for HTML delivery.	January 2, 2020