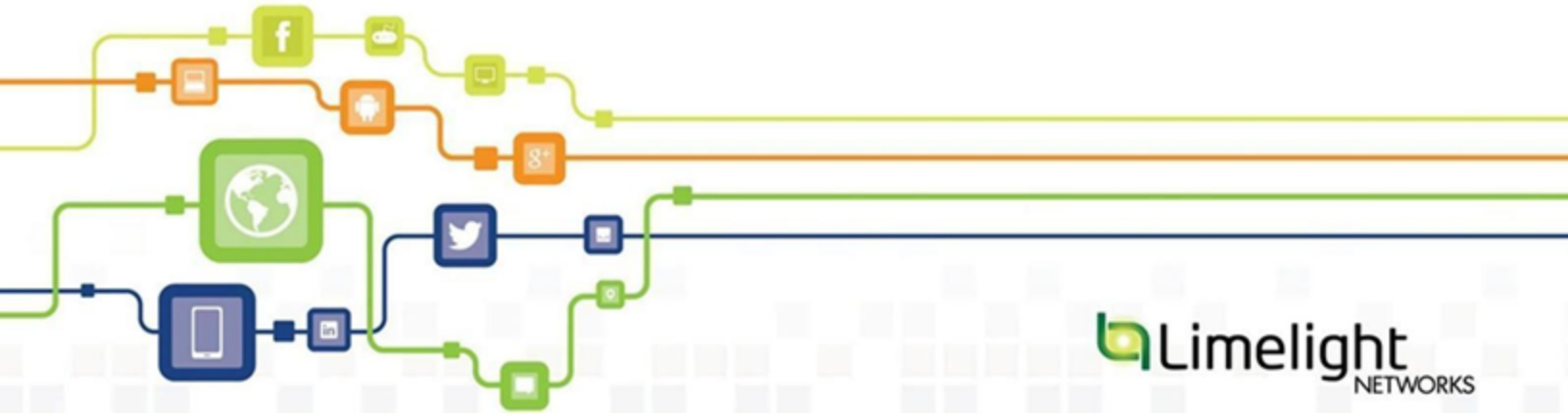

Limelight Control

Control Portal Configuration REST API User Guide

Revised 3/15/2019

Table of Contents

Revision History

Date	Summary of Changes	Document Version
2017-03-15	Original preview publication	v1
2018-02-07	Removed Option Sets; other minor changes	v2
2018-06-08	Added new *Control REST APIs Common Structure* section	v3

About This Document

This document describes the Configuration API, which customers can use to programmatically manage Content Delivery Services and Service Instances. The APIs herein provide functionality that is roughly equivalent to the configuration capabilities available in Limelight Control, the web-based customer portal.

Using the APIs in this document users can:

- Retrieve existing Content Delivery Service Profile items associated with a given shortname (also known as an Account[1]).
- Retrieve existing manual and user-managed Content Delivery Service Instance items associated with a given shortname.
- Retrieve all configuration options valid for use with given shortname and Service Profile.
- Create new Content Delivery Service Instance in the context of a Service Profile.
- Update existing Content Delivery Service Instances.
- Delete (disable) existing Content Delivery Service Instances. (You can later re-enable a Content Delivery Service Instance by updating it.)

Notes:
All the functionality described above is dependent on the API user having the necessary authorization. For example, to view a Content Delivery Service Instance in a shortname, the user must be authorized to view and manage Content Delivery Service Instances within the context of the shortname. See Create and Authorize API Users for more information.

During the Limited Availability (LA) phase of the Configuration API, any configurations created with the API can be modified only with the APIs and not with *Limelight Control*. However, users can view the configurations in *Limelight Control*.

[1]An Account is an identifier that links one or more services (and their configurations) to a customer. A customer can have multiple Accounts.

Control REST APIs Common Structure

`Control` `API`s are `RESTful` `API`s that leverage `HTTP`'s well-defined and consistent message envelope. `RESTful` architectures are designed for resource passing by placing method information in the method, scoping information in the path, client information in the headers, and the representation of the resource in the body.

The elements of the `HTTP` request are:

- Method
- `URI` path and parameters
- Header
- Representation of resource in the body or document

The elements of the `HTTP` responses are:

- Response code
- Header
- Representation of resource in the body or document

`HTTP` Requests

`URI` Path and Parameters

The general form of the `URL` for requests is:

```
https://host:port/${api-name}/${version}/${resource-name}/${sub-level}
/${resourceId}?${params}
```

where:

- **`host`** and **`port`** - define the host and port where the application lives (**`port`** is optional)
- **`api-name`** - `API` name (for example, `purge-api`)
- **`version`** - the `API` version for this feature (e.g., `v1`)
- **`resource-name`** - name of the resource requested (e.g., request)
- **`sub-level`** - optional level to further specify the requested resource
- **`resourceId`** - resource identifier, if applicable for the method
- **`params`** - additional parameters for pagination, search, authentication, etc.

> **Note:** Unless otherwise noted, arguments are case-sensitive.

`API` Version

The `REST APIs` are subject to version control. The version number of an `API` appears in its `URI`. For example, this `URI` structure requests version 1 of an `API`:

```
https://host:port/api-name/v1/resource-names
```

Notes:

- The API version number is an integer with *v* prefix, such as v1 or v2.
- The API version is independent of the release number.
- The API version may or may not change with a new release. The API version number changes only when updates to the API break the API contract, requiring changes in the code that uses the API. A change to the API does not necessarily require a change to the API version number.
- For multiple API versions, each version will ideally support at least two API versions: the latest API version and the previous API version.

Media Type

Some Limelight REST API requests accept resources, and most Limelight REST API responses return resources. These resources can be in either JSON or XML format; the default is JSON. The media type of the request resource must be specified in the Content-Type header (*application/json* or *application/xml*). The media type of the response resource is specified in the URI (responseType=json or responseType=xml) or in the *Accept* header (application/json or application/xml).

Parameters

URI parameters can be for pagination, search, authentication, and/or specific parameters for an API resource.

Authentication

The Limelight REST APIs use a combination of symmetric key cryptography and Hashed Message Authentication Code (HMAC) for message authentication and user identification.

To secure all calls to the API, an HMAC digest signature is applied to each request by using the following authentication headers:

- ***X-LLNW-Security-Principal*** - name of user performing the request. Services look up shared keys by *username* to authenticate a message. Since shared keys are stored on a per-user basis, to impersonate another user, an attacker would have to know both the username and the shared key for that user.
- ***X-LLNW-Security-Timestamp*** - request time in milliseconds. Prevents replay attacks. If the timestamp is more than *X* seconds old (usually 300), the message expires, and an error code is returned. *Note*: System clock skew minimization is an important consideration for message expiration.
- ***X-LLNW-Security-Token*** - MAC hash-generated with the user's shared key. It is calculated based on data that is sent to the server. This token is generated twice: once by the client and once by the server to compare with the one passed by the client. If the token provided by the client matches the token generated by the server, the message is authentic. Shared key is a large unique key created for use with the "HmacSHA256" MAC algorithm. Control maintains a unique and enciphered shared key for every user in the system. It is stored in HEX format and should be decoded to ASCII before usage (see code samples below). Users may access or regenerate this key at any time by using tools in Control under *My Settings > Edit My Profile*.
  - *X-LLNW-Security-Token* is formed by applying a MAC digest for the "data string" (for example, REQUEST_METHOD + URL + QUERY_STRING [if present] + TIMESTAMP + REQUEST_BODY [if present]).

Java Sample Code for HMAC Generation

```
import java.io.IOException;
import java.io.UnsupportedEncodingException;
```

```
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import org.apache.commons.codec.DecoderException;
import org.apache.commons.codec.binary.Hex;
public class HMACGenerationExample {
    private static String HMAC_ALGORITHM = "HmacSHA256";
    public static void main(String[] args) throws IOException,
InterruptedException, InvalidKeyException, NoSuchAlgorithmException,
DecoderException {
        String sharedKey = "your_key";//what you see in Control on Edit My
Profile page
        String url = "<api base url>"; //
example: "https://control.llnw.com/traffic-reporting-api/v2"
        String queryString = "<query string parameters>"; //example:
"shortname=bulkget&service=http&reportDuration=day&startDate=2012-01-01"
        String postData = "{param1: 123, param2: 456}";
        byte[] rawHmac = generateMac(sharedKey, "GET", url, queryString,
postData);
        String hmac = new String(Hex.encodeHex(rawHmac));
        System.out.println(hmac);
    }
    private static byte[] generateMac(String sharedKey, String httpMethod,
String url, String queryString, String postBody) throws
NoSuchAlgorithmException, InvalidKeyException, IllegalStateException,
UnsupportedEncodingException, DecoderException {
        byte[] decodedSharedKey = Hex.decodeHex(sharedKey.toCharArray());
        long timestamp = System.currentTimeMillis();
        String dataString;
        if (queryString == null) {
            dataString = httpMethod.toUpperCase() + url + timestamp;
        } else {
            dataString = httpMethod.toUpperCase() + url + queryString +
timestamp;
        }

        if (postBody != null) {
            dataString = dataString + postBody;
        }
        SecretKeySpec keySpec = new SecretKeySpec(decodedSharedKey, HMAC_
ALGORITHM);
```

```
        Mac mac = Mac.getInstance(HMAC_ALGORITHM);
        mac.reset();
        mac.init(keySpec);
        return mac.doFinal(dataString.getBytes("UTF-8"));
    }
}
```

Python Sample Code for HMAC Generation

```
import hashlib
import hmac
import time
try: import simplejson as json
except ImportError: import json
class HMACSample:
    def generateSecurityToken(self, url, httpMethod, apiKey,
queryParameters=None, postData=None):
            timestamp = str(int(round(time.time()*1000)))
            datastring = httpMethod + url
            if queryParameters != None : datastring += queryParameters
            datastring += timestamp
            if postData != None : datastring += postData
            token = hmac.new(apiKey.decode('hex'), msg=datastring,
digestmod=hashlib.sha256).hexdigest()
            return token
if __name__ == '__main__':
    apiEndpoint = "<api base url>"
    #example: "https://control.llnw.com/traffic-reporting-api/v2"

    #what you see in Control on Edit My Profile page#
    apiKey = "your_key";

    queryParameters = "<query string parameters>"
    #example: "shortname=bulkget&service=http&reportDuration=day&startDate=2012-
01-01"

    postData = "{param1: 123, param2: 456}"
    tool = HMACSample()
    hmac = tool.generateSecurityToken(url=apiEndpoint, httpMethod="GET",
queryParameters=queryParameters, postData=postData, apiKey=apiKey)
    print json.dumps(hmac, indent=4)
```

Request Header

The *Accept* HTTP Header can specify the response format. The default is JSON.

```
Accept=application/json
```

The *Content-Type* HTTP Header can specify the format of the request body. This header is required for all resources that require a request body.

```
Content-Type=json
```

The HTTP *X-LLNW-Security-Token, X-LLNW-Security-Principal, X-LLNW-Security-Timestamp* headers form the authentication envelope.

```
X-LLNW-Security-Token=<mac>
   X-LLNW-Security-Principal=<username>
   X-LLNW-Security-Timestamp=<now in milliseconds>
```

Representation of Resource in the Body or Document

Some APIs expect a payload of information in JSON format in the request body for PUT or POST methods. This payload is known as the body or document of the request and is a representation of the resource for the API. The format must specified with the *Content-Type* header.

HTTPS Responses

A **response header**, returned for each API request, contains:

- one of the HTTP Response Status Codes
- the returned media type in the *Content-Type* header (for example, application/json;charset=UTF-8)

The response body, returned for each API request, contains either:

- a status entity or
- a response entity (representation of the resource)

The format of the response body is determined by the *responseType* parameter or *Accept* HTTP Header in the request.

HTTP Response Status Codes

HTTP Status Code	Name	Description
5xx	Server-Side Error	Any server-side error
200	OK	Request was processed as expected.

HTTP Status Code	Name	Description
`201`	`Created`	Request created an entity.
`202`	`Accepted`	Request was processed as expected.
`304`	`Not Modified`	The requested resource has not been modified, and the client's cached representation is still valid.
`400`	`Bad Request`	The request could not be understood due to bad syntax.
`401`	`Unauthorized`	Client is not authenticated or does not have sufficient permission to perform this request. Similar to `403 Forbidden` but specifically for use when authentication is possible but has failed or has not yet been provided.
`403`	`Forbidden`	The request was a legal request, but the server is refusing to respond to it. Unlike a `401 Unauthorized` response, authenticating will make no difference.
`404`	`Not found`	The requested resource could not be found at this location (`URI`), but may be available again in the future. Subsequent requests by the client are permissible.

Configuration API Entities

Content Delivery Service Profiles

A Service Profile is a template specifying all of the available configuration options for a specific Service. Service Profiles are used to create new Service Instances (see Service Instances) with guidance on which options are required, defaulted, and optionally allowed.

A Service Profile:

- Has a name and description
- Has an associated shortname[1]
- For basic Content Delivery services, can contain up to four Protocol Sets, each of which have a unique combination of source and published protocols, possible protocols being HTTP and HTTPS
- For HTTP Chunked Streaming (HTTPCS) services, will contain a root profile, and up to four pairs of child profiles - a pair of chunked and manifest profiles for each of the 4 supported video formats - and each of these root and child profiles can contain up to 4 Protocol Sets as described above for Delivery.
- Has a Service Key (or in the case of HTTPCS, a ServiceKey per distinct root or child profile) that defines the product reflected by the Service Profile

[1]Some Service Profiles are not associated with a shortname. Known as *Global Service Profiles*, you can use for them for any Service Instance by providing, in the Service Instance, options that are allowed by the Service Profile and the shortname referenced in the Service Instance.

Each Protocol Set contains:

- Default options
- Required options
- Allowed options

See Configuration Options and Option Sets for details.

Here is part of a Service Profile definition that illustrates the items discussed in this section:

```
{
    "serviceProfileName": "Example_Test_Profile",
    "description": "Example default profile",
    "protocolSets": [
        {
            "publishedProtocol": "http",
            "sourceProtocol": "http",
            "requiredOptions": [],
            "requiredOptionSets": [],
            "defaultOptions": [],
            "allowedOptions" : []
        }
```

```
    ],
    "serviceKey": {
        "name": "delivery"
    },
    "publishedUrlPath": "/pub",
    "sourceUrlPath": "/pub"
}
```

Configuration Options

A Configuration Option is one of the choices for a specific Configuration Setting; for example, `404_backup_tcp_host`. Each of the objects returned from a `GET` call to `/configoption/shortname/{shortname}/svcProf/{svcProfileName}` is a configuration option.

The configuration options types are *default*, *required*, and *allowed*, all of which are defined in a Service Profile. A Service Instance (see Service Instances) that uses the Service Profile must meet the option requirements defined in the Service Profile:

- The caller **must** supply *Required options* in the Service Instance.
- The caller **may** provide *Default options* in the Service Instance but **does not have** to.
  - The API obtains *Default options* from the Service Profile if the caller does not supply the options in the Service Instance.
- If the *Allowed options* list is not empty, the caller **is limited** to supplying, in the Service Instance, only the *Allowed options*.

The validation process ensures that the Service Instance fulfills the option requirements.

Content Delivery Service Instances

A Service Instance is a configuration of behavior for a specific published URL associated with a specific Account. A single Service Instance may contain different configuration options for different protocol settings (that is, different options for HTTP vs HTTPS for the same published URL). Service Instances can be viewed in the Configure section of Limelight Control. A Service Instance is an actual configuration specification and:

- References a Service Profile
- Has published and source host paths
- Has a Service Key that must reflect the key defined in the Service Profile
- For basic Content Delivery services, has up to two Protocol Sets
- For HTTP Chunked Streaming (HTTPCS) services, will contain a root instance, and up to four pairs of child instances - a pair of chunked and manifest instances for each of the 4 supported video formats - and each of these root and child profiles can contain up to 2 Protocol Sets as described above for Delivery.

Each Protocol Set contains:

- Published and source ports
- Options and option sets as allowed by the Service Profile

Here is part of a Service Instance definition that illustrates the items discussed in this section:

```
{
  "serviceProfileName": "Example_Test_Profile",
  "protocolSets": [
    {
      "publishedProtocol": "http",
      "publishedPort": 80,
      "sourceProtocol": "http",
      "sourcePort": 8880,
      "options": []
    }
  ],
  "publishedHostname": "public.example.com",
  "sourceHostname": "origin.example.com",
  "publishedUrlPath": "/pub",
  "sourceUrlPath": "/pub",
  "serviceKey": {
    "name": "delivery"
   }
}
```

General API Information

Base URL

All endpoints in this document are relative to `https://apis.llnw.com/config-api/v1/`

Notes:

- The previous hostname, api.lldns.net, has been replaced by apis.llnw.com, which does not require IP whitelisting.
- All calls are made over SSL (HTTPS) and require TLS v1.1 at a minimum.

HTTP Methods

Available HTTP methods are GET, POST, PUT, and DELETE.

You can retrieve (GET) all entities. Create (POST), update (PUT) and delete (DELETE) operations are available only for Service Instances .

Request and Response Bodies

Only POST and PUT requests require a request body; GET requests do not.

All request bodies and all response bodies are represented in JSON.

Object Version

The Configuration API retains revision information on each entity. This information is updated each time the given entity is manipulated by a caller.

```
"revision": {
  "createdBy": "aUser",
  "createdDate": 1483960729773,
```

```
  "versionNumber": 1
}
```

Each time the API updates an object, the `versionNumber` member is modified accordingly.

Field References

To do any of the following activities, you need to understand references to fields in a Content Delivery Service Instance or Service Profile:

- **Sorting**: you specify the sort field for sorting results. (See Sorting.)
- **Filtering results from a GET call**: you build filters, specifying field values you want to filter on. (See Retrieve All Content Delivery Service Profiles for a Shortname and Retrieve All Manually Maintained Configurations for a Shortname.)
- **Troubleshooting Validation errors**: you determine where an error occurred in a Service Profile or Service Instance. (See Understanding Validation Errors.)

All field references are relative to `body`, the root element.

Use Path “dot notation” dot notation” to reference fields; for example the following references the `publishedProtocol` field: `body.protocolSets.publishedProtocol`.

Zero-based index dot notation is used to access the elements of any array such as `protocolSets`. So `protocolSets.0` references the first element, `protocolSets.1` references the next element, and so on.

When building filters, you can omit the array element number to cause the API to consider all elements in the array. Example: `body.protocolSets.defaultOptions.name=refresh_min` will return all Content Delivery Service Profiles that contain `refresh_min` as a default option.

Paging

GET requests that return multiple object instances (for example multiple Content Delivery Service Profiles or Service Instances) allow you to retrieve large result sets in small chunks to reduce the bandwidth required for calls. Objects are returned in a `results` array. Paging information (`size`, `offset`, `page`, `total`) and sorting information (`sort` and `sort.dir`) are also returned:

```
{
  "results": [
    {
      <first object>
    },
    {
      <next object>
    },
       . . .
```

```
  ],
  "size": "1",
  "offset": 0,
  "page": 1,
  "total": 1,
  "sort": "_id",
  "sort.dir": "ascending"
}
```

For information about the `sort` and `sort.dir` members, see Sorting.

Use the following query parameters to control paging:

- `size`: The number of results you want the server to return for each query. Example: `/svcprof/delivery/shortname/{shortname}?size=10`
  Note that in this example at most 10 records will be returned. If only five match, for example, only five will be returned.
  If you pass the value `all` for the `size` parameter, the API returns all results. Example: `/svcprof/delivery/shortname/{shortname}?size=all`
  If you omit the size parameter, the value defaults to 50
- `page`: Based on the total number of results possible, the server uses this value to return the appropriate result subset. For example if there are a total of 35 results possible and you set `size=10` and `page=3`, then page 3 contains results 21 through 30 and page 4 contains results 31 through 35.

The response body includes the following members:

- `size`: Value of the size parameter that you passed.
- `offset`: Offset from the first result possible. Example: if you have retrieved page three out of three possible results, offset will be set to 2. The `offset` calculation is: `offset = (page - 1) * size`
- `page`: Page number of the results.
- `total`: Total number of results that can be returned from your query, not taking into account paging and sizing. You can use this value to adjust your size and page values.

Here are example paging results from a request with `?size=5&page=2` as a query string:

```
"size":5,"offset":5,"page":2,"total":13,"sort":"_id","sort.dir":"ascending"
```

To use paging to return results, choose initial values for `size` and `page` and make calls until an empty results array is returned like so: `"results": []`

For example to return one record per page, set `size` and `page` to 1, then make successive calls, incrementing `page` by 1 until the `results` array is empty. Of course you can adjust `size` and `page` values during the calling sequence to account for any changes to network responsiveness.

Sorting

Results from GET calls are by default sorted in ascending order on the universally unique ID (uuid) of the desired object. If necessary, you can specify another sort key and sort direction using these query parameters:

Parameter Name	Description
sort	Sort key; name of field to sort on.
sort.dir	Sort direction. Valid values: • ascending • asc • descending • desc

For example, the following sample sorts the results in descending order by the published protocol field.

```
?sort=body.protocolSets.publishedProtocol&sort.dir=desc
```

You can combine paging and sorting query parameters:

```
?size=3&sort=body.protocolSets.publishedProtocol&sort.dir=desc
```

Your sort parameters are included in the response body:

```
"sort":"body.protocolSets.publishedProtocol","sort.dir":"descending"
```

Note: If you make an error in your sort field reference, the sort does not work but you are not notified of the error.

See Field References for more information about referencing fields.

Prerequisites

This section explains activities that must be completed before you can begin using the Limelight Control Configuration API.

Create and Authorize API Users

Before you can use the APIs described in this document, someone in your organization with Company Admin privileges must, using Limelight Control, create a special application user[1] for you, assigning the "API user" role to that user. The admin will make settings that will determine the shortnames in which you can view configurations. The company admin will also work with your company's Limelight contact who will ensure that you have create/update/delete permissions for the shortnames you will be working with.[2]

Authorization requirements for each API capability are listed under "Required Security Configuration" in the API detail tables in Service Profiles, Configuration Options and Option Sets, and Service Instances.

[1]This is the user that you will use to submit API calls via a client application.

[2]Company Admins and non-admin users will not be able to access the APIs.

After your user is created and all permissions are set, you can obtain your API shared key.

Obtain Your API Shared Key

You need an API shared key to authenticate prior to using the API.

To obtain your shared key, log into Limelight Control. Then select *My Account* in the **Welcome** drop-down menu. From there go to the *API Shared Key* section and click **Show my Shared Key**.

Authenticating explains how to use your shared key.

Authenticating

The RESTAPIs use a combination of symmetric key cryptography and Hashed Message Authentication Code (HMAC) for message authentication and user identification.

To secure all calls to the API, you apply an HMAC digest signature to each request using the following authentication headers:

Header	Description
X-LLNW-Security-Principal	Name of API user performing the request. Services look up shared keys by username to authenticate a message. Since shared keys are stored on a per-user basis, to impersonate another user, an

Header	Description
	attacker would have to know both the username and the shared key for that user.
X-LLNW-Security-Timestamp	Request time in milliseconds. Prevents replay attacks. If the timestamp is more than X seconds old (usually 300), the message expires, and an error code is returned. Note: System clock skew minimization is an important consideration for message expiration.
X-LLNW-Security-Token	MAC hash-generated with the user's shared key. It is calculated based on data that is sent to the server. This token is generated twice: once by the client and once by the server to compare with the one passed by the client. If the token provided by the client matches the token generated by the server, the message is authentic. The shared key is a large unique key created for use with the "HmacSHA256" MAC algorithm. Limelight Control maintains a unique and enciphered shared key for every user in the system. This key is stored in HEX format and should be decoded to ASCII before usage. Users may access or regenerate this key at any time by using tools in Limelight Control. Create the HMAC Digest explains how to create the value for the X-LLNW-Security-Token header.

Create the HMAC Digest

You use the HMAC digest as the value for the `X-LLNW-Security-Token` header.

1. Obtain your shared key and convert it to ASCII.
2. Using your converted shared key as the key, apply a MAC digest to the following:

   REQUEST_METHOD + URL + QUERY_STRING [if present] + TIMESTAMP + REQUEST_BODY [if present]).

   > **Note:** If your request includes a query string, do not include the “?” that marks the start of the query string.

Examples for creating the HMAC digest in Java and Python are in the "Control REST APIs Common Structure" section of the *Control Reporting REST API User Guide*.

High-Level API Tasks

This section provides high-level information about tasks you can perform with the Configuration API. Details about specific calls needed are referenced in each task.

Retrieve an Entity

1. For requests that return more than one entity instance, optionally determine paging and sorting parameters. See Paging and Sorting.
2. Determine the appropriate endpoint, HTTP method, and any query parameters you can use. See:
   - Retrieve All Content Delivery Service Profiles for a Shortname
   - Retrieve a Specific Content Delivery Service Profile
   - Retrieve All Configuration Options for a Shortname and Content Delivery Service Profile
   - Retrieve All Manually Maintained Configurations for a Shortname
   - Retrieve All Content DeliveryService Instances for a Shortname
   - Retrieve a Specific Content DeliveryService Instance
3. Depending on the call, determine the associated shortname.
4. Make the call. See Make an API Call.

Determine Configuration Option Requirements Needed When Creating or Updating an Entity

1. Optionally determine paging and sorting parameters. See Paging and Sorting.
2. Determine the associated shortname and Content Delivery Service Profile.
3. Determine the appropriate endpoint and HTTP method. See Retrieve All Configuration Options for a Shortname and Content Delivery Service Profile
4. Make the call. See Make an API Call.

Validate a Content Delivery Service Instance

1. While in the process of creating or updating a Content DeliveryService Instance, construct the appropriate JSON request message.
2. Determine the appropriate endpoint and HTTP method.

3. Make the call. See Make an API Call.

For both steps 1 and 2, see Validate a Content Delivery Service Instance.

Create a Content Delivery Service Instance

1. Determine the Content Delivery Service Profile that will provide the appropriate configuration options for the Content DeliveryService Instance. See Retrieve an Entity.
2. Construct the appropriate JSON request message. See Create a Content Delivery Service Instance and Content DeliveryService Instance Schema.
3. Validate the message (if appropriate). See Validate a Content Delivery Service Instance.
4. Determine the appropriate endpoint and HTTP method. See Create a Content Delivery Service Instance.
5. Make the call. See Make an API Call.

Update a Content Delivery Service Instance

1. Determine the specific Content DeliveryService Instance that you want to update. See Retrieve an Entity.
2. Construct the appropriate JSON request message with the needed changes.
3. Validate the message (if appropriate). See Validate a Content Delivery Service Instance.
4. For the update operation, determine the appropriate endpoint and HTTP method. See Update a Specific Content Delivery Service Instance
5. Make the call. See Make an API Call.

Delete a Content Delivery Service Instance

1. Determine the entity that you want to delete by using the GET method along with the appropriate end point. See Retrieve an Entity.
2. Determine the appropriate endpoint and HTTP method. See Delete a Specific Content Delivery Service Instance
3. Make the call. See Make an API Call.

Disable a Content Delivery Service Instance

1. Determine the specific Content DeliveryService Instance that you want to disable. See Retrieve an Entity.
2. Determine the appropriate endpoint and HTTP method. See Update a Specific Content Delivery Service Instance
3. Make the call. See Make an API Call.

Make an API Call

1. Get your shared key from Limelight Control. (One-time only.) See Obtain Your API Shared Key.
2. Generate a security token and populate request headers. See Authenticating.
3. Using the language and framework of your choice, set up a REST call.
4. Make the call, get the HTTP response code, and parse the JSON or HTML results. See Response Object

Samples for response message details.

5. If you are using paging parameters in conjunction with a GET call, keep calling until you have retrieved all results. See Paging.

REST API Operations

Service Profiles

Summary of Service Profile Capabilities

HTTP Method	URI Template	Description
GET	/svcprof/delivery/shortname/{shortname} and /svcprof/delivery/shortname/{shortname}/search	Retrieve all Content Delivery Service Profiles for a shortname. Query parameters are supported on the `/search` endpoint.
GET	/svcprof/delivery/shortname/{shortname}/{id}	Retrieve a specific Content Delivery Service Profile by ID within the context of a shortname.

Retrieve All Content Delivery Service Profiles for a Shortname

URI Template	Two templates are available: /svcprof/delivery/shortname/{shortname} /svcprof/delivery/shortname/{shortname}/search See Description for details.
Description	Depending on the URI template, you have two capabilities: `/svcprof/delivery/shortname/{shortname}`: Use this to retrieve all Content Delivery Service Profiles for a shortname. `/svcprof/delivery/shortname/{shortname}/search`: Use this to retrieve all Content Delivery Service Profiles for a shortname and also filter on field values via a query string. For example, you might want to retrieve a Content Delivery Service Profile with a specific name. See Query terms for details. When you use a query string, the API performs a validation to ensure that the field reference is valid.
Required Security Configuration	User must be authorized to view all Content Delivery Service Profile items for the shortname. The Content Delivery Service Profile must be enabled for the

<table>
<tr><td></td><td>shortname.</td></tr>
<tr><td>Pageable/Sortable?</td><td>Yes. See Paging and Sorting.</td></tr>
<tr><td>HTTP Method</td><td>GET</td></tr>
<tr><td>Query terms</td><td>Using <code>/svcprof/delivery/shortname/{shortname}/search</code>, you can add a query string. For example, this request returns an object with a specific Content Delivery Service Profile name:
<code>/search?body.serviceProfileName=LLNW_LargeObject</code>

You can also filter on fields nested at any location, using dot notation to specify the “path” to the field. Example:
<code>/svcprof/delivery/shortname/{shortname}/
search?body.protocolSets.0.defaultOptions.name=refresh_min</code>
See Field References for additional information about referencing fields.

You can also include pagination and sorting parameters with the /search endpoint:
<code>/search?body.serviceKey.name=delivery&size=5
&sort=body.serviceProfileName&sort.dir=desc</code>
See Paging and Sorting for details.</td></tr>
<tr><td>Request Content-Type</td><td>application/json</td></tr>
<tr><td>Request Body</td><td>NA</td></tr>
<tr><td>Response Status Codes</td><td>
<table>
<tr><th>Response Type</th><th>Description</th></tr>
<tr><td>200: OK</td><td>Content Delivery Service Profiles retrieved.</td></tr>
<tr><td>400: Bad Request</td><td>Wrong data (such as invalid shortname) provided.</td></tr>
<tr><td>403: Forbidden</td><td>User does not have permissions to the shortname.</td></tr>
<tr><td>404: Invalid URL Path</td><td>Invalid path component in URL.</td></tr>
<tr><td>404: API Is Down</td><td>Not found. API is down.</td></tr>
<tr><td>429: Too Many Requests</td><td>Request exceeded configured rate (one request per second).</td></tr>
</table>
</td></tr>
<tr><td>Response Content-Type</td><td>application/json</td></tr>
</table>

Retrieve a Specific Content Delivery Service Profile

URI Template	svcprof/delivery/shortname/{shortname}/{id}
Description	Retrieve a specific Content Delivery Service Profile by ID within the context of a shortname.
Required Security Configuration	User must be authorized to manage all Content Delivery Service Profile items for the shortname or at least view the configurations.
Pageable/Sortable?	No
HTTP Method	GET
Query terms	N/A
Request Content-Type	application/json
Request Body	NA
Response Status Codes	

Response Type	Description
200: OK	Content Delivery Service Profile retrieved.
400: Bad Request	Wrong data (invalid Content Delivery Service Profile ID) provided.
403: Forbidden	User does not have permissions to the shortname.
404: Invalid URL Path	Invalid path component in URL.
404: Configuration Not Found	Content Delivery Service Profile does not exist.
404: API Is Down	Not found. API is down.
429: Too Many Requests	Request exceeded configured rate (one request per second).

Response Content-Type	application/json

Configuration Options and Option Sets

Summary of Configuration Options/Option Sets Capabilities

HTTP Method	URI Template	Description
GET	/configoption/shortname/{shortname}/svcProf/{svcProfileName}	Retrieve all Configuration Options valid for use with a shortname and Content Delivery Service Profile.

Retrieve All Configuration Options for a Shortname and Content Delivery Service Profile

URI Template	/configoption/shortname/{shortname}/svcProf/{svcProfileName}
Description	Retrieve all Configuration Options valid for use with a shortname and Content Delivery Service Profile.
Required Security Configuration	User must be authorized to view and manage all Content DeliveryService Instances in the shortname.
Pageable/Sortable?	Yes. See Paging and Sorting.
HTTP Method	GET
Query terms	N/A
Request Content-Type	N/A
Request Body	N/A
Response Status Codes	**Response Type** / **Description** 200: OK — Configuration Options retrieved. 200: Empty Results List — • No valid options for the shortname / Content Delivery Service Profile combination. • Invalid shortname provided. 400: Bad Request — Wrong data (invalid Content Delivery Service Profile ID) provided. 403: Forbidden — User does not have permissions to the shortname or Content Delivery Service Profile.

	Response Type	**Description**
	404: API Is Down	Not found. API is down.
	429: Too Many Requests	Request exceeded configured rate (one request per second).
Response Content-Type	application/json	

Service Instances

Summary of Service Instance Capabilities

HTTP Method	URI Template	Description
GET	/svcinst/delivery/manual/shortname/{shortname}	Retrieve all Configurations valid for use with a shortname and Content Delivery Service Profile, that are maintained outside the scope of the API (manual rewrites and rewrites maintained via Limelight Control).
GET	/svcinst/delivery/shortname/{shortname}	Retrieve all API-managed Content DeliveryService Instances valid for use with a shortname.
GET	/svcinst/delivery/{id}	Retrieve a specific API-managed Content DeliveryService Instance.
POST	/svcinst/delivery/validate	Validate a Content DeliveryService Instance based on the Content Delivery Service Profile referenced in the Content DeliveryService Instance.
POST	/svcinst/delivery	Create a Content DeliveryService Instance based on Content Delivery Service Profile referenced within the Content Delivery Service Instance.
PUT	/svcinst/delivery/{id}	Update a specific Content DeliveryService Instance.
DELETE	/svcinst/delivery/{id}	Delete a specific Content DeliveryService Instance.

Retrieve All Manually Maintained Configurations for a Shortname

URI Template	/svcinst/delivery/manual/shortname/{shortname}
Description	Retrieve all manually maintained configurations for a shortname. The returned configurations are represented as Content Delivery Service Delivery Instances. You can also include a query string to filter on field values. See Query terms for details. Configurations returned from this endpoint include configurations created both via

<table>
<tr><td></td><td>Limelight Control and the Configuration API. For configurations created in Limelight Control, the value of the <code>serviceProfileName</code> member is <code>readOnlyControl</code>. For manual rewrites, the value is <code>migration</code>.</td></tr>
<tr><td>Required Security Configuration</td><td>User must be authorized to view or manage all Content DeliveryService Instances in the shortname.</td></tr>
<tr><td>Pageable/Sortable?</td><td>Yes. See Paging and Sorting.</td></tr>
<tr><td>HTTP Method</td><td>GET</td></tr>
<tr><td>Query terms</td><td>Use query terms to filter on field values with capabilities for exact and partial matches:
Exact match: match on entire field value. Example: Search for a configuration with accountId=2431:
GET /svcinst/delivery/manual/shortname/{shortname}?accountId=2431
Partial match: Combine “like_” with a field name. Example:
GET /svcinst/delivery/manual/shortname/{shortname}?like_accountId=24
This query returns all configurations with account ID that contains “24”</td></tr>
<tr><td>Request Content-Type</td><td>application/json</td></tr>
<tr><td>Request Body</td><td>N/A</td></tr>
<tr><td>Response Status Codes</td><td><table>
<tr><th>Response Type</th><th>Description</th></tr>
<tr><td>200: OK</td><td>All manually maintained configurations retrieved.</td></tr>
<tr><td>200: Empty Results List</td><td>Invalid field reference while attempting to filter on a field value.</td></tr>
<tr><td>403: Forbidden</td><td>Invalid shortname provided or user does not have permissions to view configurations in the shortname.</td></tr>
<tr><td>404: Invalid URL Path</td><td>Invalid path component in URL.</td></tr>
<tr><td>404: API Is Down</td><td>Not found. API is down.</td></tr>
<tr><td>429: Too Many Requests</td><td>Request exceeded configured rate (one request per second).</td></tr>
</table></td></tr>
<tr><td>Response Content-Type</td><td>application/json</td></tr>
</table>

Retrieve All Content Delivery Service Instances for a Shortname

URI Template	/svcinst/delivery/shortname/{shortname}
Description	Retrieve all API-managed Content DeliveryService Instances valid for use with a

	shortname.
Required Security Configuration	User must be authorized to view and manage all Content DeliveryService Instances in the shortname.
Pageable/Sortable?	Yes. See Paging and Sorting.
HTTP Method	GET
Query terms	N/A
Request Content-Type	application/json
Request Body	N/A
Response Status Codes	**Response Type** / **Description** 200: OK — Content Delivery Service Instances retrieved. 403: Forbidden — User does not have permissions to view Content DeliveryService Instances in the shortname, or invalid shortname provided. 404: Invalid URL Path — Invalid path component in URL. 404: API Is Down — Not found. API is down. 429: Too Many Requests — Request exceeded configured rate (one request per second).
Response Content-Type	application/json

Retrieve a Specific Content Delivery Service Instance

URI Template	svcinst/delivery/{id}
Description	Retrieve a specific API-managed Content Delivery Service Instance by ID.
Required Security Configuration	User must be authorized to view and manage all Content DeliveryService Instances in the shortname related to the Content DeliveryService Instance. The related Content Delivery Service Profile must be configured for the shortname or company.
Pageable/Sortable?	No
HTTP Method	GET
Query terms	N/A
Request Content-Type	application/json

<table>
<tr><td>Request Body</td><td>N/A</td></tr>
<tr><td>Response Status Codes</td><td>
<table>
<tr><th>Response Type</th><th>Description</th></tr>
<tr><td>200: OK</td><td>Content Delivery Service Instance retrieved.</td></tr>
<tr><td>403: Forbidden</td><td>User does not have permissions to view configurations in the shortname.</td></tr>
<tr><td>404: Configuration Not Found</td><td>Content Delivery Service Instance does not exist.</td></tr>
<tr><td>404: Invalid URL Path</td><td>Invalid path component in URL.</td></tr>
<tr><td>404: API Is Down</td><td>Not found. API is down.</td></tr>
<tr><td>429: Too Many Requests</td><td>Request exceeded configured rate (one request per second).</td></tr>
</table>
</td></tr>
<tr><td>Response Content-Type</td><td>application/json</td></tr>
</table>

Validate a Content Delivery Service Instance

<table>
<tr><td>URI Template</td><td>/svcinst/delivery/validate</td></tr>
<tr><td>Description</td><td>Validate a Content DeliveryService Instance on the basis of the Content Delivery Service Profile referenced in the Content Delivery Service Instance. The API applies all validation logic to the Content Delivery Service Instance, ensuring that the Content Delivery Service Profile exists and that all of the request payload complies with the Content Delivery Service Profile’s requirements:
All required fields are present
All options are allowed for use by the particular customer
All options are valid and allowed for the user as per the Content Delivery Service Profile
Note:
Validation is designed for verifying the structure of a Content DeliveryService Instance JSON document when creating or updating a Content DeliveryService Instance. Validation is best used in an interactive context such as a user creating the Content DeliveryService Instance in a user interface where the validation can provide immediate feedback. If you are creating/updating a Content DeliveryService Instance from a backend component where the JSON document is created programmatically, an individual validation step is not necessary because the requests to create and update the entity include a validation step.</td></tr>
<tr><td>Required Security Configuration</td><td>User must be authorized to view and manage all Content DeliveryService Instances in (1) the Content Delivery Service Profile referenced in the request body and (2) the shortname related to the Content Delivery Service Profile.</td></tr>
</table>

<table>
<tr><td>Pageable/Sortable?</td><td>No</td></tr>
<tr><td>HTTP Method</td><td>POST</td></tr>
<tr><td>Query terms</td><td>None</td></tr>
<tr><td>Request Content-Type</td><td>application/json</td></tr>
<tr><td>Request Body</td><td>Request Body Sample</td></tr>
<tr><td>Response Status Codes</td><td>
<table>
<tr><th>Response Type</th><th>Description</th></tr>
<tr><td>200: OK</td><td>No API errors occurred during validation and the Content Delivery Service Instance passed validation.

See Response Body for additional details.</td></tr>
<tr><td>200: OK</td><td>No API errors occurred during validation but the Content Delivery Service Instance failed validation.

See Response Body for additional details.</td></tr>
<tr><td>400: Bad Request</td><td>Bad data provided, such as JSON syntax error.</td></tr>
<tr><td>403: Forbidden</td><td>User does not have permission to validate Content DeliveryService Instances in the related shortname.</td></tr>
<tr><td>404: API Is Down</td><td>Not found. API is down.</td></tr>
<tr><td>404: Invalid URL Path</td><td>Invalid path component in URL.</td></tr>
<tr><td>429: Too Many Requests</td><td>Request exceeded configured rate (one request per second).</td></tr>
<tr><td>500: Internal Error</td><td>Missing required schema element.</td></tr>
</table>
</td></tr>
<tr><td>Response Content-Type</td><td>application/json</td></tr>
<tr><td>Response Body</td><td>
Note:
The 200 status code is independent of whether the Content Delivery Service Instance itself passed validation.

The success member indicates success or failure of a validation request:
• "success": true: No validation errors were found. The API returns the Content DeliveryService Instance object with modifications: if there were defaultOptions specified in
</td></tr>
</table>

	the referenced Content Delivery Service Profile that were not in the request payload, they will appear in the `options` array of the returned Content DeliveryService Instance. The "revision" block (see Object Version) will also be generated and returned as part of the response. • **"success": false**: Validation errors detected. The API returns errors in an errors object. The Understanding Validation Errors section describes the `errors` object so you can successfully parse the object to discover error details.
Validation Details	1. The API validates the input schema as a first step. 2. The `publishedUrlPath` can contain a regex, no regex, or a combination of regex and literals. Any regex patterns you include must be in Limelight's approved regex pattern database. (Contact Limelight Support for information about the approved regex database.) • If you provide a regex in the `publishedUrlPath`, you must also provide a wrapper object {} to wrap the regex that you are trying to configure as the `publishedUrlPath`. Example: `"publishedUrlPath": "/{(.*somePublishedPath))}"` • Any request that is sent with a regex that is not wrapped in {} will be rejected. • To validate the regex, the API parses all regular expressions present in the `publishedUrlPath`, looks for matches in Limelight's approved regex database, then saves the data in the event of a successful validation. Otherwise, the API rejects the request with 400 status code. (Contact Limelight Support for information about the approved regex database.) 3. If callers provide invalid or unrecognized options or option sets, the API will return a 200 status code along with "`success" : false` in the response body.

Create a Content Delivery Service Instance

URI Template	/svcinst/delivery
Description	Create a Content DeliveryService Instance based on a specific Content Delivery Service Profile (referenced in the `serviceProfileName` member) passed in the request payload. The API will obtain default options from either: • The Content DeliveryService Instance passed in the request, if the caller provides the options • From the Content Delivery Service Profile default configuration if the caller does not provide the options in the Content DeliveryService Instance passed in the request The API then persists the Content DeliveryService Instance and creates a corresponding Edge Prism configuration. **Note:** To create a Content DeliveryService Instance you must first determine the Content Delivery Service Profile that will provide the appropriate configuration options. **Note:** A recommended best practice is to create the Content DeliveryService Instance with its `isEnabled` member set to `false`. (This creates the Content DeliveryService Instance but does not push it to the edge.) Then have members of your organization perform a peer review on the Content DeliveryService Instance. After the peer review is complete and the Content DeliveryService Instance is approved, you can update the Content DeliveryService Instance, setting `isEnabled` to `true`. See Update a Specific Content Delivery Service Instance.
Required Security Configuration	User must be authorized to view and manage all Content DeliveryService Instances in (1) the Content Delivery Service Profile referenced in the request body and (2) the shortname related to the Content Delivery Service Profile.
Pageable/Sortable?	No
HTTP Method	POST
Query terms	N/A
Request Content-Type	N/A
Request Body	Request Body Sample
Response Status Codes	**Response Type** / **Description** 200: OK / Content DeliveryService Instance created successfully. The API returns the Content DeliveryService Instance definition.

	<table><tr><th>Response Type</th><th>Description</th></tr><tr><td>400: Already Exists</td><td>An identical object already exists. Objects are considered “identical” if they have the same publishedURL value, which is a concatenation of the publishedHostname and publishedUrlPath values.</td></tr><tr><td>400: Bad Request</td><td>Bad request, wrong data provided.</td></tr><tr><td>400: Validation Error</td><td>Content Delivery Service Instance failed validation.</td></tr><tr><td>403: Forbidden</td><td>User does not have permission to create Content Delivery Service Instances in the related shortname.</td></tr><tr><td>404: API Is Down</td><td>Not found. API is down.</td></tr><tr><td>404: Invalid URL Path</td><td>Invalid path component in URL.</td></tr><tr><td>429: Too Many Requests</td><td>Request exceeded configured rate (one request per second).</td></tr></table>
Response Content-Type	application/json
Validation Details	1. The API validates the input schema as a first step. 2. The `publishedUrlPath` can contain a regex, no regex, or a combination of regex and literals. Any regex patterns you include must be in Limelight’s approved regex pattern database. (Contact Limelight Support for information about the approved regex database.) • If you provide a regex in the `publishedUrlPath`, you must also provide a wrapper object `{}` to wrap the regex that you are trying to configure as the `publishedUrlPath`. Example: `"publishedUrlPath": "/{(.*somePublishedPath))}"` • Any request that is sent with a regex that is not wrapped in `{}` will be rejected. • To validate the regex, the API parses all regular expressions present in the `publishedUrlPath`, looks for matches in Limelight’s approved regex database, then saves the data in the event of a successful validation. Otherwise, the API rejects the request with 400 status code. (Contact Limelight Support for information about the approved regex database.) 3. If callers provide invalid or unrecognized options or option sets, the API will return a 200 status code along with `“success” : false` in the response body.

Update a Specific Content Delivery Service Instance

URI Template	/svcinst/delivery/{id}

<table>
<tr><td>Description</td><td>Update a specific Content DeliveryService Instance identified by its ID. Each update results in an updated <code>revision</code> member within the Content DeliveryService Instance. See Object Version for more information.

Notes:
A specialized use of this endpoint and HTTP method is to disable a Content DeliveryService Instance. You do so by setting the Content DeliveryService Instance's <code>isEnabled</code> attribute to <code>false</code>. Doing so removes the configuration from the edge but keeps the configuration available so you can later re-enable it. Disabling a Content DeliveryService Instance is useful and recommended when you are working with test configurations or troubleshooting the Content DeliveryService Instance.

To enable a Content DeliveryService Instance after disabling it, you must update it, setting its <code>isEnabled</code> attribute to <code>true</code>.</td></tr>
<tr><td>Required Security Configuration</td><td>User must be authorized to view and manage all Content DeliveryService Instances in (1) the Content Delivery Service Profile referenced in the request body and (2) the shortname related to the Content Delivery Service Profile. User must also be authorized to view and manage the Content Delivery Service Profile and shortname that were configured in the original Content DeliveryService Instance.</td></tr>
<tr><td>Pageable/Sortable?</td><td>No</td></tr>
<tr><td>HTTP Method</td><td>PUT</td></tr>
<tr><td>Query terms</td><td>N/A</td></tr>
<tr><td>Request Content-Type</td><td>N/A</td></tr>
<tr><td>Request Body</td><td>The request body must be the JSON returned from a GET call to retrieve the entity, modified according to your requirements.

Request Body Sample</td></tr>
<tr><td>Response Status Codes</td><td>
<table>
<tr><th>Response Type</th><th>Description</th></tr>
<tr><td>200: OK</td><td>Content DeliveryService Instance updated and a copy is returned to the caller.</td></tr>
<tr><td>400: Validation Error</td><td>Content Delivery Service Instance failed validation.</td></tr>
<tr><td>403: Forbidden</td><td>User does not have permissions to update configurations in the shortname.</td></tr>
<tr><td>404: API Is Down</td><td>Not found. API is down.</td></tr>
<tr><td>404: Invalid URL Path</td><td>Invalid path component in URL.</td></tr>
</table>
</td></tr>
</table>

	Response Type	**Description**
	404: Not Found	Invalid Content DeliveryService Instance ID.
	429: Too Many Requests	Request exceeded configured rate (one request per second).
Response Content-Type	application/json	

Delete a Specific Content Delivery Service Instance

URI Template	/svcinst/delivery/{id}
Description	Delete a specific Content DeliveryService Instance from the edge, identified by its ID. You should delete a Content Delivery Service Instance only if you are sure that you are done with a published URL and do not expect to use it anymore. The deletion is actually a "soft" deletion because the Content DeliveryService Instance itself is not deleted; it is simply removed from the edge. You can only delete a Content DeliveryService Instance once. The first time you attempt to delete a Content DeliveryService Instance, the response is 200 OK and the API returns a copy of the deleted object in the response body. Subsequent attempts result in a 404 response with a response body similar to `404 Content Delivery Service Instance Already Deleted`.
Required Security Configuration	User must be authorized to view and manage all Content DeliveryService Instances in the shortname related to the Content DeliveryService Instance. The related Content Delivery Service Profile must be configured for the shortname or company.
Pageable/Sortable?	No
HTTP Method	DELETE
Query terms	N/A
Request Content-Type	application/json
Request Body	N/A

Response Status Codes	**Response Type**	**Description**
	200: OK	Content DeliveryService Instance deleted. The API returns a copy of the deleted Content DeliveryService Instance.
	403: Forbidden	User not allowed to delete Content Delivery Service Instances in the

	Response Type	**Description**
		shortname.
	404: Already Deleted	Content DeliveryService Instance already deleted.
	404: API Is Down	Not found. API is down.
	404: Invalid URL Path	Invalid path component in URL.
	404: Not Found	Invalid Content DeliveryService Instance ID.
	429: Too Many Requests	Request exceeded configured rate (one request per second).
Response Content-Type	application/json	

Appendix A - Configuration Options

This section describes the options that can be submitted in the `options` array in the `protocolSets` object of a request.

Examples

In this example, the `negative_ttl` option is requested with a value of `30` seconds:

```
{
    "uuid": "62119e2f-488c-4d18-ad99-ece63fbe341c",
    "isLatest": true,
    "isEnabled": true,
    "body": {
          "sourceHostname": "source.host.com",
          "serviceProfileName": "docExample",
          "protocolSets": [
             {
                "optionSets": [],
                "publishedProtocol": "http",
                "sourceProtocol": "http",
                "options": [
                   {
                      "name":"negative_ttl",
                      "parameters":[
                         30
                      ]
                   }
                ],
...
```

In this example, the `allow_methods` option is requested with multiple parameters:

```
...
                "options": [
                   {
                      "name":"request_send_header",
                      "parameters":[
                         "header_name","header_value"
                      ]
                   }
                ],
...
```

Options

Option Category	Name	Description	Parameters
Content acquisition	404_backup_origin	Specify a new origin base URL for the request on 404. The original origin base URL is still used as the cache key. This allows the preservation of cache keys while changing origins.	<URL>
Content acquisition	404_backup_tcphost	404 backup TCP host specifies the host to connect to in the event that the origin or a host specified in a force_tcphost rewrite returns a 404 HTTP status code. You can also specify an optional port number.	<FQDN or IP>[:port]
Content acquisition	404_fallbackurl	404 fallback URL specifies a URL to be used for content retrieval instead of sending a 404 response code. The original request is reissued to the fallback URL with any modifications still in place. The rewrite is used to provide a custom "not found" message or to provide instructions to retry the request. Any URL used for this rewrite must match its own rewrite within the EdgePrism configuration.	<full URL>
Cache key manipulation	add_trailing_slash	Add a trailing slash to source URLs if a browser issues a GET request without one.	
Cache key manipulation	add_trailing_slash_regex	Add a trailing slash to source URLs if a browser issues a GET request without one; based on provided regex.	<regex>
Media delivery	allow_flvstart	Implements FLV Seek, which allows a user to scrub forward or backward in an FLV file during progressive download. Used when a customer wants users to have streaming-like experiences but with HTTP rather than streaming.	
Security	allow_methods	Takes a comma-separated list of method names, -- e.g. get,-head,options. These are not case-sensitive. Requests with any other method are denied. Note that any request with a body is treated as a POST.	<method>,<method2>,...
Security	allow_methods_deny_code	The return code used when a request is denied because it doesn't match the list in an allow_methods rewrite. If this rewrite is not present but the allow_methods rewrite is, the return code for 'denied' is 400.	<HTTP code>
Media delivery	allow_moovseek	allow_moovseek implements H.264 Seek, which allows a user to scrub forward or backward in an MP4 file during progressive download. Used when a customer wants users to have streaming-like experiences but with HTTP rather than streaming.	
Media delivery	allow_rate_limit	Sets the rate limit to your lowest possible bit rate measured in kBytes per second. This is not a cap: it's a minimum.	<number>
Cache control	always_refresh	Initiates a freshness check for every object request unless the object age is less than the time specified with a refresh_absmin rewrite option. NOTE: Use with caution as this may cause origin servers to become overloaded.	
Cache control	assume_cachable	On a cache miss, assumes the object is NOT cacheable, so if another request comes in for the same object, another request is made to the	

		origin. This can cause multiple requests unnecessarily going back to the origin. With this option enabled, the object is assumed to be cacheable.	
Cache control	assume_ cachable_ no_refresh	Same as the assume_cachable option without refresh checking	
Content acquis-ition	backup_ origin	Specifies a new origin base URL for the request on failure (defined the same way as in backup_tcphost). The original origin base URL is still used as the cache key to allow preservation of cache keys while changing origins.	<URL>
Content acquis-ition	backup_ tcphost	Backup TCP host specifies the host to connect to in the event that the origin or the force_tcphost fails. You can optionally specify a port number.	<FQDN or IP>[:port]
Security	bad_req_ redirect	If the request is not valid (e.g., check failures from hashsecret, referer, ipgeo_match, ip= match), EdgePrism responds with a 302 HTTP redirect to this specified URL.	<URL>
Security	block_ referer	This feature is used to prevent hotlinking by listing domains of known offending sites. When a Referer header matches a domain in the list then the request will be denied. However, this check is not strict. Should a request not contain a Referer header or should the URL contained in the header be difficult to parse, a request is allowed. If stricter checking is desired, see the strict_referer_checking option.	<domain.-com>,<domain2.com>
Cache control	cache_ gen-erated_ responses	For HTTP responses generated dynamically, origins often do not supply cache control headers (e.g. Cache-Control, Expires, or Last-Modified). With this rewrite option, EdgePrism can still consider such responses cacheable. The cache_generated_responses rewrite option accepts a TTL used to control refreshing of objects that match this rewrite.	<seconds>
Cache key manip-ulation	cache_ tag_allow_ request	Allow setting the cache tag from a client request. This is intended for use with ICS.	<1, 0, or -1>
Cache key manip-ulation	cache_ tag_allow_ response	Allow setting the cache tag from an origin response.	<1, 0, or -1>
Cache key manip-ulation	cache_ tag_ header_ name	Specifies the name of the cache tag header	<header_name>
Cache key manip-ulation	cache_ tag_ header_ strip	Strips the cache tag header when returning response to clients.	<1, 0, or -1>
Header control	cdn_ ident_style	Supplies the extra header X-Cache-Request: limelight when making the request to the origin server.	<1>
Request handling	chunked_ requests	Allows or disallows chunked requests (e.g., requests with a "Transfer-Encoding: chunked" header). "chunked_requests 1" allows chunked requests for the rewrite "chunked_requests -1" disallows chunked requests for the rewrite "chunked_requests 0" is a no-op. A disallowed	<1, 0, or -1>

		chunked request fails with a "411: Length Required" error message. Note that the discard_request_body option does not work with chunked messages. Also note that this option implies allowing HTTP/1.1 POST requests, which allow the client to send an Expect header. This may cause the origin to legitimately reply with a 100 Continue response. EdgePrism does not support 100 Continue headers. To manage 100 Continue headers, couple this rewrite with the remove_request_header rewrite with the value "Expect" ('remove_request_header Expect').	
Compression	compress_mimetype_regex	Specifies the type of content to be compressed via the Content-Type header. The regular expression is matched against the header value returned from the origin.	<regex>
Compression	compress_regex	Specifies the type of content to be compressed via the file extension. The regular expression is matched against the published URL of the request to determine if compression should be applied.	<regex>
Header control	cors_acac	A response header that can be sent in CDN responses to the client. The rewrite can be set to control whether EdgePrism sends the Access-Control-Allow-Credentials (ACAC) header in the response to the client, and what the ACAC header value will be each time. The service can be configured to behave in one of two ways: 1. Do not return an ACAC header. If the origin returns one, delete it from the response sent to the client. 2. Do nothing. If the origin returns this header, simply pass it through to the client. The header value will be set to "true". Note: If the ACAC header is "true", the ACAO header cannot be "*".	<acac>
Header control	cors_acao	Configures the Access-Control-Allow-Origin (ACAO) header in the response to the client, and specifies what the ACAO header value will be each time. The service can be configured to behave in one of these ways: 1. Do not return an ACAO header. If the origin returns one, delete it from the response sent to the client. 2. Do nothing. If the origin returns this header, simply pass it through to the client. 3. Specify that the he ACAO header value will be "*". 4. Use the same value as that in the Origin: header, as the value of the ACAO response header.	<acao>
Header control	cors_aceh	Controls the Access-Control-Expose-Headers (ACEH) header in the response to the client, as well as the value of the ACEH header. The ACEH header is a comma-separated list of header names. Note: The headers in this list should not be any of these simple headers: Cache-Control, Content-Language, Content-Type, Expires, Last-Modified, Pragma.	<list>
Header control	cors_origin	A request header that can be sent in CDN requests to the origin. A specific value can be set as the origin header, which EdgePrism then passes to the origin for cache fill. If the client sends an origin header in the initial request, this setting overrides the client-provided header.	<origin>
Header control	cust_dbg_access	Traces a URL to a matched rewrite. Debugging and troubleshooting header. Information is added to the header in X-LLNW-Dbg-Cdn-Rewrite.	<list of keywords in desired dbg headers>
Header control	cust_dbg_oldsecret	Adds a secret used for debugging and troubleshooting headers.	<secret>
Header control	cust_dbg_secret	Provides a new secret used for debugging and troubleshooting headers.	<secret>

Security	disable_ mv_set_ cookie	Removes MediaVault cookie (a Set-Cookie: header) from the response sent to the client. MediaVault cookies received from the client are still processed and validated.	
Response handling	disable_ reply_ send_ chunked	Overrides reply_send_chunked. When this option is specified on a rewrite, EdgePrism does not send chunked transfer-encoded responses to HTTP 1.1 clients.	
Header control	discard_ request_ body	When enabled, POST bodies are accepted but then discarded, behaving as if the request were a GET. This is used for clients that use POST to grab an object rather than GET.	
Content acquis- ition	dont_ send_ origin_ maxage	EdgePrism sends a 72-hour Cache-Control: max-age=259200 header on every request to origin if a max-age does not already exist. Disables automatically including max-age if one does not exist.	
Cache control	exclude_ query_ terms	Affects which parts of a query string are used to generate cache keys. This option specifies which query term parameters should be excluded in the cache key if present in a query string.	<comma separated terms>
Content acquis- ition	fallbackurl	Fallback URL specifies a URL to return instead of sending a 503 (Service Unavailable) or 504 (Gateway Timeout) response. The 503 or 504 may have been generated by the origin, but could also be intern- ally generated by EdgePrism should a connection failure to origin occur.	<full URL>
Cache key manip- ulation	fold_case	Allows the case of URLs in the cache to be folded. The given case is still used to make the request to the origin, but requests for the same object with different case will be served out of cache.	
Cache control	force_ nostore	Forces content not to be cached/stored. Also causes EdgePrism to cease negative caching.	
Cache control	force_ nostore_ by_status	Forces content not to be cached/stored. Also causes EdgePrism to cease negative caching.	<comma separated values>
Content acquis- ition	force_ origin	Specifies a new origin base URL for the initial request. The original origin base URL is still used as the cache key. This allows the pre- servation of cache keys while changing origins. For example, it would allow an https:// published with http:// cache key while maintaining https:// connectivity to origin, thereby ensuring that the http:// and https:// rewrite pairs "share" cache keys, reducing, and potentially halving, cache utilization, without risking redirect loops if a URL is requested using http and redirected by the origin to https.	<URL>
Content acquis- ition	force_ server_ header	Allows "Server" header to be set	<server name>
Content acquis- ition	force_ tcphost	When present, EdgePrism uses the hostname specified in the force_ tcphost rewrite option as an origin instead of the one provided in the source URL in a CDN rewrite. The host header is still populated with the host portion of the source URL. A common use of force_tcphost is to allow customization of the host header by specifying the desired value in the source URL, and the origin TCP host via the force_tcphost rewrite option. You can specify an optional port number.	<FQDN or IP>[:port]

Response handling	genreply	Generates a response using a defined response code	<HTTP Code>
Response handling	genreply_ options	When genreply_options is included on a rewrite, EdgePrism does not contact the origin, but instead generates an HTTP reply with the specified HTTP return code.The genreply_options rewrite option only takes effect when the HTTP request method is OPTIONS.	<HTTP Code>
Request handling	grab_ range_ from_url	Allows for a range request to be parsed from the URL in the format of http://pub.example.com/file.ext/range/#-#. The '/range/#-#' is translated into an actual range request to the origin.	
Com- pression	gziponly	Allows clients that request only gzip to receive compressed content (via the Accept-Encoding header). Major HTTP clients send both gzip and deflate.	
Security	hash_ https_ compat	Combined with hashsecret option, hashes of urls differ between when http:// and https:// are used. Customers who are hashing https:// URLs with http:// by mistake, this option will fall back to the http:// hash and still succeed if https:// hash fails. (Example: https://ll-salestr.h-s.llnwd.net/o1/s/welcome.txt?h=40a4d1121975ade41d84e5b6a7d48394 would be the correct hash, but http://ll-salestr.h-s.llnwd.net/o1/s/welcome.txt?h=5ba9d776f4909d2a294d902eecc4bc5a will succeed as well.')	
Security	hashpath	Set the hash path string in this option.	
Security	hashsecret	MediaVault is a server-side URL authentication service. It should be noted that MediaVault is not a replacement for DRM and should not be associated with user authentication in any way. MediaVault's main purpose is to assist customers in securing their content from unau- thorized viewing. The hashsecret option is where the hash character string should be set.	<character string>
Security	hash- secret_ cookie	Standard MediaVault parameters are URL-based. The new scheme places them within HTTP cookies after the initial manifest request. Fur- thermore, EdgePrism manages sequences of requests by adding incre- mental expiration to the cookies.	
Security	hash- secret_ deny_ code	Allows changing of return HTTP code from MediaVault default rejection code of 400 to another value.	<HTTP Code>
Security	hash- secret_ except_ regex	Allows MediaVault configuration to apply to negatively matched URL strings by regular expression.	<regex>
Security	hash- secret_ first_bytes	Allow MediaVault to be applied only if the request is withing the first X number of bytes.	<byte size>
Security	hash- secret_ only_ regex	Allows MediaVault configuration to apply to positively matched URL strings by regular expression.	<regex>

Security	hash-secret_queryter-m_list	Allows MediaVault configuration to specify which query terms are to be included in the authentication. The first three parameters accept 0 or 1. The queryterm-list parameter accepts one or more comma-separated strings.	<val-only> <order> <exclude> <queryterm-list>
Security	hash-secret_redirect_param	Sets a custom URL that is sent via a redirect when a MediaVault authorization fails. This single parameter is removed from the MediaVault query string and placed in the Location: header in a redirect. The "code" parameter expects 301 or 302. The "location" parameter requires a full URL, and "parameter" accepts the name of the parameter as a string.	<code> <location> <parameter>
Security	hash-secret_url_parse_regex	Allows MediaVault configuration to parse (extract) a portion of the URL, for use in the hash calculation, using a regular expression. Set the fail-flag to 1 to fail the request immediately if the regex does not match, or to 0 to continue trying to validate with normal MediaVault authentication.	<fail-flag> <regex>
Header control	hitmiss_trigger_header	Allows for a specified request header to be added in order to return HIT/MISS information for an object. Does not track peer hierarchy HIT/MISS information like X-LDebug does.	<header name>
Cache control	ignore_all_vary	When specified, causes EdgePrism to disregard any vary header in making cacheability determinations.	
Cache control	ignore_bad_status	Ignores bad status codes from the origin (403, 404, 403, etc.) which would normally cause content to be uncached to allow object to remain in cache. Stale content may be served as a result.	
Cache control	ignore_cacheability_header_value	Used to specify a list of vary headers to ignore when determining cacheability beyond Accept-Encoding. Will set ignore_vary if the token is 'vary'.	<header-name> <value> [,<value>]...
Cache control	ignore_nocache	Causes these no-cache header values to be ignored when determining cacheability. Cache-control: no-cache Cache-control: no-store Cache-control: private Pragma: no-cache	
Cache control	ignore_qt_ip	Disables MediaVault IP address checks.	
Cache control	ignore_vary	Ignores a specific Vary http header from the origin. The second argument is the specific header being returned from the origin. Setting ignore_cacheability_header value set ignore_vary if the token is 'vary'.	<comma separated vary header values>
Content acquisition	inm_to_ims	Enable basic If-None-Match support. If-None-Match on an incoming request will be matched against a cached object's ETag header to enable sending 304 Not Modified. Upstream refresh checks, partial caching requests, etc., will be done without If-None-Match and will use If-Modified-Since when appropriate. NOTE - this does not give full ETag support and does NOT work with Vary, multiple ETags per object, Range requests, If-Match, If-Range, ...	
Compression	intgz	Enables gzip to be done at the edge instead of sending the request to the specified source in the rewrite. Only files between 50b and 256k are compressed by default.	
Compression	intgz_level	Controls the gzip compression level (0 to 9).	<number>

Com-pression	intgz_ maxsize	Controls the maximum file size in bytes that will be gzipped on the edge; default is 256k.	<number>
Com-pression	intgz_non-cacheable	Enables non-cacheable responses to be gzipped on the edge. It gzips on the fly even when the content cannot be re-used. All POST responses are compressed.	
Security	ipgeo_ deny_ code	Specifies the response HTTP code from default rejection code of 400 to another value when geo match fails.	<http code>
Security	ipgeo_ match	Specifies whitelist or blacklist geo matching.	<comma separated country codes and/or whitelists>
Cache key manip-ulation	keep_qt_ prefix	This rewrite option allows query terms with a common prefix to be retained in the URL for caching content when strip_query_terms_store is enabled.	<character string>
Cache control	keep_ query_ terms	Specifies which query term parameters should be included in a cache key when used in conjunction with strip_query_terms_store or another rewrite that strips query terms (e.g. MediaVault's hashsecret).	<comma separated terms>
Logging	log_ request_ header	When the log_request_header rewrite option is present, all EdgePrisms in a hierarchy will log the value of a specified client request header.	<header name>
Logging	log_ request_ header2	When the log_request_header rewrite option is present, all EdgePrisms in a hierarchy will log the value of a specified client request header.	<header name>
Logging	log_ request_ header3	When the log_request_header rewrite option is present, all EdgePrisms in a hierarchy will log the value of a specified client request header.	<header name>
Logging	log_ request_ header4	When the log_request_header rewrite option is present, all EdgePrisms in a hierarchy will log the value of a specified client request header.	<header name>
Logging	log_ request_ header5	When the log_request_header rewrite option is present, all EdgePrisms in a hierarchy will log the value of a specified client request header.	<header name>
Request handling	max_ bytes_post	The max_bytes_post configuration sets the maximum number of bytes allowed for a given HTTP POST or PUT, preventing larger requests from being processed. When a POST or PUT request exceeds this limit, EdgePrism returns a 413 "Request Entity Too Large". The default limit for POST or PUT requests is 500MB.	<Byte size>
Media delivery	moov_ atom_ maxsize	Allows a per-rewrite maximum moov atom size. EdgePrism does not perform moov-seek on media files with moov atoms larger than this size. Setting this to a larger value overrides the global setting. The default (and minimum) is 10MB; the maximum is 50MB.	<size>
Cache control	negative_ abs_ttl	Specifies a TTL value in seconds for EdgePrism to negatively cache response from customer origin and overrides origin cache control headers. The current system default negative TTL is 10 seconds.	<seconds>
Cache control	negative_ ttl	Specifies a TTL value in seconds for EdgePrism to negatively cache response from customer origin and overrides origin cache control headers. The current system default negative TTL is 10 seconds.	<seconds>

Compression	nocompress_regex	If a published URL does not match the regular expression, EdgePrism is allowed to request compressed content from the origin, as well as compress content in the CDN.	<regex>
Security	old-hashsecret	Provides the ability to have a second hash secret. This rewrite can be used when transitioning from one hashsecret to another while maintaining older hashsecret links in circulation. The argument is the shared secret.	<character string>
Content acquisition	origin_conn_timeout	Timeout setting before an attempt is made to a backup_tcphost or 'fail' the request.	<seconds>
Content acquisition	origin_keepalive_timeout	The maximum amount of time EdgePrism maintains an idle connection to an origin.	<seconds>
Content acquisition	origin_max_lifetime	The amount of time a connection may be kept open when an HTTP transaction to origin has completed and the time has passed. The connection to the origin will be closed.	<seconds>
Content acquisition	origin_reply_timeout	The number of seconds EdgePrism waits for a reply after submitting a request to an origin. The minimum is 3 seconds; the default is the read_timeout global setting.	<seconds>
Cache control	partial_caching	Enables partial caching for an entire rewrite	
Cache control	partial_caching_regex	Enables partial caching for a rewrite based on the defined regular expression match	<regex>
Header control	persistent_connection_off	Enables multiple HTTP requests on the same client TCP connection, improving performance by reducing the overhead of establishing TCP connections. This behavior is on by default, and this option disables it on a per-rewrite basis.	
Cache control	preserve_etag	When preserve_etag is enabled, for GET requests (not range requests), EdgePrism does not strip ETag headers for cached objects, and will return an ETag header for requested objects.	
Cache control	preserve_etag_all	When enabled for range requests, EdgePrism does not strip ETag headers for cached objects, and will return an ETag header for requested objects.	
Media delivery	qtalias_start_fs	Creates an alias for the MediaVault parameter "start=" which maps to "fs=".	
Media delivery	qtalias_start_ms	Creates an alias for the MediaVault parameter "start=" which maps to "ms=".	
Security	query_term_alias	Provides the ability to 'map' request-specific queryterm names to EdgePrism queryterm names since they are quite often the same thing but different in name. Note that this option does not modify the request URL.	<customer-qt>:<our-qt>,<customer-qt2>:<our-qt2>[, ...]
Security	query_term_ignore	Provides a list of queryterms to be ignored by EdgePrism. The queryterms are not evaluated or validated by queryterm processing. This option should be used when a queryterm is needed outside of EdgePrism. Note that any queryterms appearing before the specified queryterm to ignore are still included in the hash calculation but will	<qt1,qt2[, ...]>

		not be processed in any special way. Note that this option does not modify the request URL.	
Cache control	quick_ abort_min	Sets the minimum amount of data in KB that we need to deliver to cache the file. General usage is -1 to cause us to cache no matter when the client disconencts.	<number>
Cache control	range_ offset_limit	Sets an upper limit on how far into the the file a range request may be to cause EdgePrism to prefetch the whole file. If beyond this limit, then EdgePrism forwards the range request as it is, and the result is not cached.	<number>
Request handling	range_ query_ term	Enables range requests to be set as the value of a queryterm. If the option is "range_query_term foo" then a request for ../-filename?foo=<range spec> will be handled exactly as if a header with "Range: bytes=<range spec>" had been included in the request. If a range header is present in a request, as well as a query-term range spec, the range header is ignored. But if there is no query-term range spec, that information will be logged, and any range headers present will be used normally. It should be used with one of the rewrite options strip_query_terms_store, exclude_query_terms, or keep_query_terms, so that different range requests do not get assigned different cache keys.	<query term>
Redirect	redirect	Instead of delivering content from the origin, EdgePrism can redirect the user to a particular URL. The redirect rewrite option uses the origin URL as a base to which the rest of the original request is appended. The <code> parameter may be 301 or 302, and EdgePrism uses the the rewritten URL as the Location header returned to the client.	<301 or 302>
Redirect	redirect_ host-name_ header_ regex	This option can be used to issue a redirect based on a specified value and header. In the case of a regex match of the value within the specified header, EdgePrism will issue an HTTP redirect where the Location header will contain the original url with the hostname replaced by the hostname associated with the match. If the regex starts with an exclamation point, EdgePrism will perform a negative match, where EdgePrism will only issue a redirect if there is no match.	<header name> <HTTP status code> <regular expres-sion>:<hostname> [,<regular expres-sion>:<hostname>]*
Security	referer	This feature is used to prevent hotlinking by listing allowed sites. Only requests that have a referer header matching a domain in this list will be allowed. However, this check is not strict. Should a request not contain a referer header or should the URL contained in the header be difficult to parse, a request is allowed. If stricter checking is desired, see the strict_referer_checking option. Note that a domain name can be registered and constructed to get around this (e.g. for a referer of bar.com, someone could register foobar.com.) One workaround is to configure this as "referer .bar.com", so that a period has to appear in front of the domain name to match the argument (which will not work if TLD names are used).	<domain.-com>,<domain2.com>
Cache control	refresh_ absmax	Forces EdgePrism to use an absolute maximum value before returning to origin for a freshness check. This option overrides Header settings.	<seconds>
Cache control	refresh_ absmin	Forces EdgePrism to use an absolute minimum value before returning to origin for a freshness check. This option overrides Header settings.	<seconds>
Cache control	refresh_ max	Forces EdgePrism to use a specified maximum value before returning to origin for a freshness check unless the origin specifies a value in the headers.	<seconds>

Cache control	refresh_ min	Forces EdgePrism to use a specified minimum value before returning to origin for a freshness check unless the origin specifies a value in the headers.	<seconds>
Header control	remove_ request_ header	Strips the request header(s) on requests going direct to origin.	<header_1>,<header_ 2>,...,<header_n>
Header control	remove_ response_ header	Strips the response header sent by the origin so it is not sent to the client.	<header_1>,<header_ 2>,...,<header_n>
Response handling	reply_ send_ chunked	Allows EdgePrism to send chunked transfer-encoded responses to HTTP 1.1 clients (for reference, see RFC 2616, section 3.6.1) for cache misses when an origin response does not contain a Content-Length header.	
Header control	reply_ send_ header	Sends a single custom header/value pair to client responses. Header value may be URL encoded to permit spaces. The IP address can be added into the response header by adding {IP} to the header value. The first occurrance of {IP} is replaced with the IP address of the edge-most EdgePrism server that delivered the object. The reply_send_ header option can be used multiple times on the same rewrite. In this way, more than three headers can be sent in the reply.	<header name> <header value>
Header control	reply_ send_ header_ by_ext	Allows headers to be modified based on URL path extension matches. For example, one may decide to force a specific content type for all jpeg images. Multiple header and extensions are allowed. These headers are only added for 200 and 206 HTTP response status codes.	<extension 1>:<header name 1>:<header value 1>,...,<extension n>:<- header name n>:<- header value n>
Header control	reply_ send_ header_ options	This option only takes effect when the HTTP request method is OPTIONS. This option can be specified multiple times on a rewrite. It sends a single custom header/value pair to client responses. Header value may be URL encoded to permit spaces. The IP address can be added into the response header by adding {IP} to the header value. The first occurrance of {IP} is replaced with the IP address of the edge-most EdgePrism server that delivered the object.	<header name> <header value>
Header control	reply_ send_ ipaddr	Sends a custom response header, and value is added to responses sent to clients. X-IP-Address is the header; the value is the IP address of the edge-most EdgePrism server that delivered the object.	
Header control	req_send_ client_ip	This feature specifies a header to send to the origin, containing the value of the client's IP address. This header is in addition to X-Forwarded-For, which is always provided to the origin.	<header name>
Header control	req_send_ header	Sends a single custom header/value pair to the origin on request. The header value may be URL encoded to permit spaces. req_send_ header can be used multiple times on the same rewrite. In this way, more than three headers can be sent in the request.	<header name> <header value>
<header name>	<header name>	<header name>	<header name>
Header control	req_send_ header_ ipgeo_ country	Send only the country field from the available IP geo information back to the origin via a request header.	<header name>

Header control	req_send_src_url_header	Allows EdgePrism caches to inject or overwrite a header to reflect the source URL from an associated rewrite.	<header name>
Header control	rewrite_via_header	Allows EdgePrism to rename the via header to an arbitrary name. This is typically used in situations where IIS may not compress content when it receives a via header.	<header name>
Request handling	robots_disallow_all	Generates a disallow all robots response when /robots.txt is requested. This option should only be applied to rewrites that are at the doc-root (/) level, not deeper.	
Arclight Computed Edge Policies	rules_on_any_request	Runs for every request that comes into any box in a hierarchy.	<Path to rule file> <Function name>
Arclight Computed Edge Policies	rules_on_any_response	Runs every time a response is received from an origin or a peer. Specifically, it runs the instant that HTTP headers are received, but not the body.	<Path to rule file> <Function name>
Arclight Computed Edge Policies	rules_on_client_response	Experimental stage - Runs rules upon client response.	<Path to rule file> <Function name>
Arclight Computed Edge Policies	rules_on_edge_request	Runs rules on every edge request, not peer requests.	<Path to rule file> <Function name>
Arclight Computed Edge Policies	rules_on_origin_request	Experimental stage - Runs rules upon origin request.	<Path to rule file> <Function name>
Arclight Computed Edge Policies	rules_on_origin_response	Runs every time a response is received from an origin, and not from peer	<Path to rule file> <Function name>
Security	s3_sign_basic	Support S3 authorization when using Amazon S3 storage as an origin. When EdgePrism fetches an asset from origin this option adds an S3 authorization header, which looks something like this: Authorization: AWS AKIAIQ64VUQFAC6A7WNQ:Nb-b17OOI7s6o7/bfNXrPZD2EPco=. In order to generate this header, a rewrite option supplying an access key identifier and secret access key is provided (S3 temporary credentials are not supported).	<access_key_id> <secret_access_key>
Security	s3_sign_basic_region	Supports S3 signature V4 authorization when using Amazon S3 storage as an origin. This option depends on the s3_sign_basic option; it specifies the S3 region that is serving the content.	<s3_region>

Content acquis-ition	stale_on_ not_ found_on	Normally, when EdgePrism receives a 404 response from the origin, it is passed back to the requesting client. With this rewrite, if there is cached content for a given request and the origin returns a 404, EdgePrism returns the stale content instead of issuing a 404 to the client.	
Redirect	strict_ header_ regex_ checking	The strict_header_regex_checking option specifies the parameters for the redirect EdgePrism will issue in the case where the client request does not contain the header specified in the redirect_hostname_ header_regex option.	<HTTP code> <hostname>
Cache control	strip_ query_ terms_ store	Removes any appended query terms when creating the cache key.	
Content acquis-ition	treat_ empty_ 200_as_ 404	Used when files have not successfully finished copying on origins.	

Appendix B - Request Payload Samples

Note:
The samples in this section have not been tested and will be updated in the next version of this document.

Validate a Content Delivery Service Instance

```
{
  "isLatest": true,
  "body": {
    "sourceHostname": "source.host.com",
    "serviceProfileName": "docExample",
    "protocolSets": [
      {
        "publishedPort": 8090,
        "publishedProtocol": "http",
        "sourceProtocol": "http",
        "sourcePort": 8090,
        "optionSets": [
        ],
        "options": [

        ]
      },
      {
        "publishedPort": 443,
        "publishedProtocol": "https",
        "sourceProtocol": "https",
        "sourcePort": 8443,
        "optionSets": [
        ],
        "options": [
        ]
      }
    ],
```

```
    "sourceUrlPath": "\/ttt",
    "publishedUrlPath": "\/so13",
    "serviceKey": {
      "name": "delivery"
    },
    "publishedHostname": "published.host.com"
  },
  "accounts": [
    {
      "shortname": "sample"
    }
  ]
```

Create a Content Delivery Service Instance

```
{
   "isLatest":true,
   "body":{
         "sourceHostname":"source.host.com",
         "serviceProfileName":"docExample",
         "protocolSets":[
            {
              "publishedProtocol" :  "http" ,
              "sourceProtocol" : "http",
              "optionSets":[
               ],
            }
         ],
         "sourceUrlPath":"/content/",
         "publishedUrlPath":"/content/",
         "serviceKey":{
            "name":"delivery"
         },
         "publishedHostname":"published.host.com"
   },
   "revision":{
      "versionNumber":1,
      "createdDate":1468004111992,
      "createdBy":"userName"
   },
```

```
   "accounts":[
      {
         "shortname":"sample"
      }
   ]
}
```

Update a Content Delivery Service Instance

```
{
   "uuid": "62119e2f-488c-4d18-ad99-ece63fbe341c",
   "isLatest": true,
   "isEnabled": true,
   "body": {
         "sourceHostname": "source.host.com",
         "serviceProfileName": "docExample",
         "protocolSets": [
            {
               "optionSets": [],
               "publishedProtocol": "http",
               "sourceProtocol": "http",
               "options": [
                  {
                     "name": "hashsecret",
                     "parameters": [
                        "secret"
                     ]
                  }
               ],
               "publishedPort": 8090,
               "sourcePort": 8090
            },
            {
               "optionSets": [],
               "publishedProtocol": "https",
               "sourceProtocol": "https",
               "options": [],
               "publishedPort": 443,
               "sourcePort": 8443
            }
```

```
      ],
      "sourceUrlPath": "/content",
      "publishedUrlPath": "/content",
      "serviceKey": {
         "name": "delivery"
      },
      "publishedHostname": "published.host.com"
},
"revision": {
   "versionNumber": 1,
   "createdDate": 1476729480555,
   "createdBy": "userName"
},

"accounts": [
   {
      "shortname": "sample"
   }
]
```

Appendix C - Response Object Samples

Common Responses

The responses in this section can be returned from various requests.

200 Empty results List

The `results` array is always empty. Values of other members depend on the specific request.

```
{
  "results": [],
  "size": "25",
  "offset": 0,
```

```
  "page": 1,
  "total": 0,
  "sort": "_id",
  "sort.dir": "ascending"
}
```

400 Bad Request and 403 Forbidden

The response messages for 400 and 403 are identical.

```
{
  "message":"API Request failed"
}
```

> **Note:**
> Some calls, such as Delete a Content DeliveryService Instance, return a specific JSON response for a 400 response code.

404 API Is Down

The "API Is Down" response, presented in HTML, is similar to this example:

404 Configuration Not Found

```
{
  "success": false,
  "error": {
    "code": "config.notFound",
```

```
    "message": "No config found with uuid: bd86f1ab-3b72-44d2-9fb1-
0d0f9b6d2d1a",
    "args": [
      "bd86f1ab-3b72-44d2-9fb1-0d0f9b6d2d1a"
    ]
  }
}
```

404 Invalid URL Path

The "Invalid URL Path" error is always presented in HTML.

In this example, the caller included an invalid path: `/invalid/path`. which yields the following response:

429 Too Many Requests

```
{
  "message":"Requests exceed configured rate"
}
```

500 Internal Error While Creating a Configuration or Validating a Content Delivery Service Instance

```
{
  "errorType": "internal",
  "error": "Internal error transforming config",
  "success": false
```

```
}
```

Retrieve All Content Delivery Service Profiles for a Shortname

200 OK

> **Note:**
> This sample has not been tested and will be updated in the next version of this document.


```
{
  "results": [
    {
      "uuid": "8dfd2a06-2fb8-4948-aa92-7632b943e9be",
      "isLatest": true,
      "isEnabled": true,
      "body": {
        "protocolSets": [
          {
            "publishedProtocol": "http",
            "publishedPort": 80,
            "sourceProtocol": "http",
            "sourcePort": 8090,
            "options": [],
            "optionSets": [
              "site_intgz"
            ]
          },
          {
            "publishedProtocol": "https",
            "publishedPort": 80,
            "sourceProtocol": "https",
            "sourcePort": 8090,
            "options": [],
            "optionSets": []
          }
        ],
        "serviceProfileName": "docExample",
        "serviceProfileRelativePath": "/",
        "publishedHostname": "published.host.com",
```

```
      "sourceHostname": "source.host.com",
      "publishedUrlPath": "/content/",
      "sourceUrlPath": "/content/",
      "serviceKey": {
        "name": "delivery"
      }
    },
    "revision": {
      "createdBy": "userName",
      "createdDate": 1485521510708,
      "versionNumber": 1
    },
    "accounts": [
      {
        "shortname": "sample"
      }
    ]
  },
  {
    "uuid": "5df93fac-3f2b-4948-aa92-4783ac8e6d5a",
    "isLatest": true,
    "isEnabled": true,
    "body": {
      "protocolSets": [
        {
          "publishedProtocol": "https",
          "publishedPort": 80,
          "sourceProtocol": "http",
          "sourcePort": 8090,
          "options": [],
          "optionSets": [
            "site_intgz"
          ]
        },
        {
          "publishedProtocol": "https",
          "publishedPort": 80,
          "sourceProtocol": "https",
          "sourcePort": 8090,
          "options": [],
          "optionSets": []
        }
```

```
        ],
        "serviceProfileName": "docExample",
        "serviceProfileRelativePath": "/",
        "publishedHostname": "published.host.com",
        "sourceHostname": "source.host.com",
        "publishedUrlPath": "/content/",
        "sourceUrlPath": "/content/",
        "serviceKey": {
          "name": "delivery"
        }
      },
      "revision": {
        "createdBy": "userName",
        "createdDate": 1485521510708,
        "versionNumber": 1
      },
      "accounts": [
        {
          "shortname": "sample"
        }
      ]
    }
  ],
  "size": "2",
  "offset": 0,
  "page": 1,
  "total": 564,
  "sort": "_id",
  "sort.dir": "ascending"
}
```


Retrieve a Specific Content Delivery Service Profile

200 OK

> **Note:**
> This sample has not been tested and will be updated in the next version of this document.


```
{
```

```
    "uuid": "8dfd2a06-2fb8-4948-aa92-7632b943e9be",
    "isLatest": true,
    "isEnabled": true,
    "body": {
      "protocolSets": [
        {
          "publishedProtocol": "http",
          "publishedPort": 80,
          "sourceProtocol": "http",
          "sourcePort": 8090,
          "options": [],
          "optionSets": [
            "site_intgz"
          ]
        },
        {
          "publishedProtocol": "https",
          "publishedPort": 80,
          "sourceProtocol": "https",
          "sourcePort": 8090,
          "options": [],
          "optionSets": []
        }
      ],
      "serviceProfileName": "docExample",
      "serviceProfileRelativePath": "/",
      "publishedHostname": "published.host.com",
      "sourceHostname": "source.host.com",
      "publishedUrlPath": "/content/",
      "sourceUrlPath": "/content/",
      "serviceKey": {
        "name": "delivery"
      }
    },
    "revision": {
      "createdBy": "userName",
      "createdDate": 1485521510708,
      "versionNumber": 1
    },
    "accounts": [
      {
        "shortname": "sample"
```

```
        }
      ]
    }
```

Retrieve All Configuration Options for a Shortname and Content Delivery Service Profile

200 OK

400 Bad Request

200 OK

Note:
This sample has not been tested and will be updated in the next version of this document.

```
{
  "results": [
    {
      "uuid": "1f85e410-04c5-432e-bcf4-41ae276a9914",
      "isLatest": true,
      "isEnabled": true,
      "body": {
        "optionName": "_404_backup_origin",
        "optionDetails": {
          "description": "Specify a new origin base URL for the request on 404.
The original origin base URL is still used as the cache key. This allows the
preservation of cache keys while changing origins.\n",
          "syntax": "404_backup_origin <URL>",
          "argumentList": [
            {
              "argNumber": 1,
              "name": "URL",
              "type": "Str"
            }
          ],
          "nargs": 1,
          "example1": "cdn_rewrite http:\/\/public.example.com\/
http:\/\/src.example.com\/ acct_id # 404_backup_origin
http:\/\/example3.com\/dir1\/",
```

```
          "featureCategory": "Content acquisition"
        }
      }
    },
    {
      "uuid": "44d79def-4b89-4dac-80c9-d7d93d9edbbb",
      "isLatest": true,
      "isEnabled": true,
      "body": {
        "optionName": "404_fallbackurl",
        "optionDetails": {
          "featureName": "404 Backup Handling: Alternate URL",
          "description": "404 Fallback URL specifies a URL to be used for
content retrieval instead of sending a 404 response code. The original request
is reissued to the fallback URL with any modifications still in place. The
rewrite is used to provide a custom \"not found\" message or to provide
instructions to retry the request. Any URL used for this rewrite must match its
own rewrite within the EdgePrism configuration.\n",
          "syntax": "404_fallbackurl <full URL>",
          "argumentList": [
            {
              "argNumber": 1,
              "name": "Full URL",
              "type": "Str"
            }
          ],
          "nargs": 1,
          "example1": "cdn_rewrite http:\/\/public.example.com\/
http:\/\/src.example.com\/ acct_id # 404_fallbackurl
http:\/\/pub2.example.com\/custom404.html",
          "changedEpRelease": "4.2.4.0",
          "featureCategory": "Content acquisition"
        }
      }
    }
  ]
}
```


400 Bad Request


```
{
```

```
  "success": false,
  "errorType": "validation",
  "errors": "deliverysvcProfile was not found for the deliverysvcprofName
:docExampel"
}
```

Retrieve All Manually Maintained Configurations for a Shortname

200 OK

> **Note:**
> This sample has not been tested and will be updated in the next version of this document.

This sample retrieves two configurations out of all possible configurations.

```
{
  "results": [
    {
      "body": {
        "sourceHostname": "source.host.com",
        "serviceProfileName": "migration",
        "description": "manual rewrite migration",
        "protocolSets": [
          {
            "optionSets": [
              "site_intgz"
            ],
            "publishedProtocol": "http",
            "sourceProtocol": "http",
            "options": [
              {
                "name": "force_tcphost",
                "parameters": [
                  "origin-uatdisclosuresquest.canaccordgenuity.com"
                ]
              }
            ],
          }
        ],
        "sourceUrlPath": "/",
```

```
      "publishedUrlPath": "/",
      "serviceKey": {
        "name": "delivery"
      },
      "publishedHostname": "published.host.com"
    },
    "accountId": "7300"
  },
  {
    "body": {
      "sourceHostname": "source.host.com",
      "serviceProfileName": "migration",
      "description": "manual rewrite migration",
      "protocolSets": [
        {
          "optionSets": [
            "site_intgz"
          ],
          "publishedProtocol": "http",
          "sourceProtocol": "http",
          "options": [
            {
              "name": "force_tcphost",
              "parameters": [
                "origin-www.pcsc.ca"
              ]
            }
          ],
        }
      ],
      "sourceUrlPath": "/",
      "publishedUrlPath": "/",
      "serviceKey": {
        "name": "delivery"
      },
      "publishedHostname": "published.host.com"
    },
    "accountId": "7300"
  }
],
"size": 2,
"offset": 0,
```

```
  "page": 1,
  "total": 37877,
  "sort": "_id",
  "sort.dir": "ascending"
}
```

Content DeliveryService Instance Example

The sample in this section illustrate a single Content Delivery Service Instance, which can be returned from any of these calls:

- Retrieve a specific Content DeliveryService Instance
- Create, update, or delete a Content DeliveryService Instance

> **Note:**
> This sample has not been tested and will be updated in the next version of this document.

```
{
  "uuid": "8dfd2a06-2fb8-4948-aa92-7632b943e9be",
  "isLatest": true,
  "isEnabled": true,
  "body": {
    "protocolSets": [
      {
        "publishedProtocol": "http",
        "publishedPort": 80,
        "sourceProtocol": "http",
        "sourcePort": 8090,
        "options": [

        ],
        "optionSets": [
          "site_intgz"
        ]
      },
      {
        "publishedProtocol": "https",
        "publishedPort": 80,
        "sourceProtocol": "https",
        "sourcePort": 8090,
        "options": [
```

```
        ],
        "optionSets": [

        ]
      }
    ],
    "serviceProfileName": "docExample",
    "serviceProfileRelativePath": "\/",
    "publishedHostname": "published.host.com",
    "sourceHostname": "source.host.com",
    "publishedUrlPath": "\/content\/",
    "sourceUrlPath": "\/content\/",
    "serviceKey": {
      "name": "delivery"
    }
  },
  "revision": {
    "createdBy": "userName",
    "createdDate": 1485521510708,
    "versionNumber": 1
  },
  "accounts": [
    {
      "shortname": "sample"
    }
  ]
}
```

Retrieve All Content DeliveryService Instances for a Shortname

200 OK

Note:
This sample has not been tested and will be updated in the next version of this document.

```
{
  "results": [
    {
```

```
  "uuid": "8dfd2a06-2fb8-4948-aa92-7632b943e9be",
  "isLatest": true,
  "isEnabled": true,
  "body": {
    "protocolSets": [
      {
        "publishedProtocol": "http",
        "publishedPort": 80,
        "sourceProtocol": "http",
        "sourcePort": 8090,
        "options": [],
        "optionSets": [
          "site_intgz"
        ]
      },
      {
        "publishedProtocol": "https",
        "publishedPort": 80,
        "sourceProtocol": "https",
        "sourcePort": 8090,
        "options": [],
        "optionSets": []
      }
    ],
    "serviceProfileName": "docExample",
    "serviceProfileRelativePath": "/",
    "publishedHostname": "published.host.com",
    "sourceHostname": "source.host.com",
    "publishedUrlPath": "/content/",
    "sourceUrlPath": "/content/",
    "serviceKey": {
      "name": "delivery"
    }
  },
  "revision": {
    "createdBy": "userName",
    "createdDate": 1485521510708,
    "versionNumber": 1
  },
  "accounts": [
    {
      "shortname": "sample"
```

```
        }
      ]
    }
  ],
  "size": "100",
  "offset": 0,
  "page": 1,
  "total": 564,
  "sort": "_id",
  "sort.dir": "ascending"
}
```

Retrieve a Specific Content DeliveryService Instance

200 OK

See Content DeliveryService Instance Example.

Validate a Content DeliveryService Instance

200 Successful Validation

(For a failed validation sample, see Content DeliveryService Instance Validation Error Example.)

> **Note:**
> This sample has not been tested and will be updated in the next version of this document.

```
{
  "success": true,
  "config": {
     "uuid": null,
     "isLatest": true,
     "isEnabled": true,
     "body": {
           "sourceHostname": "source.host.com",
           "serviceProfileName": "docExample",
           "protocolSets": [
              {
                 "optionSets": [],
```

```
                "publishedProtocol": "http",
                "sourceProtocol": "http",
                "options": [
                   {
                      "name": "priority",
                      "parameters": [
                         8
                      ]
                   }
                ],
                "publishedPort": 8090,
                "sourcePort": 8090
             },
             {
                "optionSets": [],
                "publishedProtocol": "https",
                "sourceProtocol": "https",
                "options": [],
                "publishedPort": 443,
                "sourcePort": 8443
             }
          ],
          "sourceUrlPath": "/ttt",
          "publishedUrlPath": "/h2/qatest",
          "serviceKey": {
             "name": "delivery"
          },
          "publishedHostname": "published.host.com"

      },
      "revision": {
         "versionNumber": 0,
         "createdDate": 1476742372893,
         "createdBy": "userName"
      },
      "status": {
         "state": "PENDING",
         "stepMap": {}
      },
      "accounts": [
         {
            "shortname": "sample"
```

```
        }
      ]
    }
}
```

Create a Content DeliveryService Instance

200 OK

400 Already Exists

200 OK

See Content DeliveryService Instance Example.

400 Already Exists

```
{
  "success": false,
  "errorType": "validation",
  "errors": {
    "body": [
      {
        "code":
"validatorInternal.deliverysvcinst.error.deliverySvcInstanceConflict",
        "message": "DeliverySvcInstance already exists with uuid",
        "args": [
          "812224cf-e430-47b9-b1aa-2b096d2d0e05"
        ],
        "type": "validator"
      }
    ]
  }
}
```

Update a Specific Content DeliveryService Instance

200 OK

See Content DeliveryService Instance Example.

Content DeliveryService Instance Validation Error Example

Validation errors can occur while validating, creating, or updating a Content Delivery Service Instance.


```
{
  "success": false,
  "errorType": "validation",
  "errors": {
    "body.protocolSets.0.source.protocol": [
      {
        "level": "error",
        "keyword": "enum",
        "message": "instance value (\"httpp\") not found in enum (possible
values: [\"http\",\"https\"])",
        "value": "httpp",
        "enum": [
          "http",
          "https"
        ],
        "schemaRef": "\/definitions\/DeliverySvcInstance\/properties
\/protocolSets\/items\/properties\/source\/properties\/protocol",
        "type": "schema",
        "code": "jsonSchema.error.enum",
        "args": [
          "[http,https]",
          "httpp"
        ]
      }
    ]
  }
}
```


Delete a Specific Content DeliveryService Instance

200 OK

404 Content Delivery Service Instance Already Deleted

200 OK

See Content DeliveryService Instance Example.

404 Content Delivery Service Instance Already Deleted


```
{
  "success": false,
  "error": {
    "code": "config.deleted",
    "message": "The config with uuid: a49e589e-f86d-4294-ab7b-59fc414577e5 has
been deleted",
    "args": [
      "a49e589e-f86d-4294-ab7b-59fc414577e5"
    ]
  }
}
```

Appendix D - Understanding Validation Errors

This section explains the structure of a validation failure response, using a simple response as a sample. This knowledge will help you to programmatically detect and understand the failure.

When a validation fails, the `success` member is set to false and errors are contained in an `errors` object in the response to the validation request. Here is a sample response in its entirety:

```
{
  "success": false,
  "errorType": "validation",
  "errors": {
    "body.protocolSets.0.sourceProtocol": [
      {
        "level": "error",
        "keyword": "enum",
        "message": "instance value (\"httpp\") not found in enum (possible
values: [\"http\",\"https\"])",
        "value": "httpp",
        "enum": [
          "http",
          "https"
        ],
        "schemaRef":
"\/properties\/protocolSets\/items\/properties\/sourceProtocol",
        "type": "schema",
        "code": "jsonSchema.error.enum",
        "args": [
          "[http,https]",
          "httpp"
        ]
      }
    ]
  }
}
```

Each error object contains a key that indicates where in the request message the error occurred. Here is the key:

```
{
  "success": false,
  "errorType": "validation",
  "errors": {
```

```
    "body.protocolSets.0.sourceProtocol":
    . . .
```

The key indicates that there is an error in the source protocol of the first protocol of the protocol sets.

See Field References for additional information.

The `message` member describes the error:

```
"body.protocolSets.0.sourceProtocol": [
      {
        "level": "error",
        "keyword": "enum",
        "message": "instance value (\"httpp\") not found in enum (possible
values: [\"http\",\"https\"])",
        "value": "httpp",
        "enum": [
          "http",
          "https"
        ]
```

The `message` member above indicates that the caller passed an incorrect value (`httpp`) and that the correct values are `http` and `https`.

The error object also points to the location in the schema where the error occurred. The `schemaRef` member provides a path-like reference to the error's location:

```
"schemaRef": "\/properties\/protocolSets\/items\/properties\/sourceProtocol",
        "type": "schema",
        "code": "jsonSchema.error.enum",
        "args": [
          "[http,https]",
          "httpp"
        ]
```

Use the reference to learn more about that part of the schema by looking at the Schema section corresponding to the entity with which you are working.

> **Note:**
> The `body` element in the `errors` object corresponds to the `properties` element in the schema.

Appendix E - Schemas

Schemas are useful both for troubleshooting (see Understanding Validation Errors) and for generating objects in the programming languages of your choice.

Schema Relationships

A Content DeliveryService Instance uses a Service Profile. Both Content DeliveryService Instances and Content Delivery Service Profiles use a Protocol Set.

Content Delivery Service Profile Schema

A Content Delivery Service Profile Schema is an abstraction of a rewrite with user-friendly field names.

```
{
  "title": "DeliverySvcProfile",
  "description": "A profile that is used for DeliverySvc affects Provisioning
API behavior for that service",
  "additionalProperties": false,
  "required": [
    "serviceProfileName"
  ],
  "type": "object",
  "properties": {
    "protocolSets": {
      "type": "array",
      "description": "Array of ProtocolSet comprising of list of possible
published protocol and sourceprotocol combination and  the various types  of
options optionsets supported for a given combination ",
      "minItems": 1,
      "maxItems": 4,
      "uniqueItems": true,
      "items": {
        "additionalProperties": false,
        "type": "object",
        "required": [
          "publishedProtocol",
          "sourceProtocol"
        ],
        "properties": {
          "publishedProtocol": {
```

```
            "description": "published Protocol ",
            "type": "string",
            "enum": [
              "http",
              "https"
            ]
          },
          "publishedPort": {
            "description": "Published  Port ",
            "type": "integer",
            "minimum": 0,
            "maximum": 65535
          },
          "sourceProtocol": {
            "description": "Source Protocol ",
            "type": "string",
            "enum": [
              "http",
              "https"
            ]
          },
          "sourcePort": {
            "description": "Source  Port ",
            "type": "integer",
            "minimum": 0,
            "maximum": 65535
          },
          "allowedOptions": {
            "description": "Rewrites options that are allowed in a service
profile",
            "type": "array",
            "items": [
              {
                "type": "string"
              }
            ]
          },
          "defaultOptions": {
            "description": "Default values that are to be applied if the rewrite
option is not supplied in the service profile",
            "minItems": 0,
            "type": "array",
```

```
            "items": [
              {
                "$ref": "#\/definitions\/option"
              }
            ]
          },
          "requiredOptions": {
            "description": "Options and their values that are required to be
provided as part of a service profile",
            "type": "array",
            "minItems": 0,
            "items": [
              {
                "type": "string"
              }
            ]
          },
          "allowedOptionSets": {
            "description": "Names of OptionSets to be allowed in this service
profile",
            "minItems": 0,
            "type": "array",
            "items": [
              {
                "type": "string"
              }
            ]
          },
          "defaultOptionSets": {
            "description": "Names of OptionSets that are default in this service
profile",
            "minItems": 0,
            "type": "array",
            "items": [
              {
                "type": "string"
              }
            ]
          },
          "requiredOptionSets": {
            "description": "Names of required\/mandatory OptionSets to be used
in this service profile",
```

```
            "minItems": 0,
            "type": "array",
            "items": [
              {
                "type": "string"
              }
            ]
          }
        }
      }
    },
    "uuid": {
      "description": "Unique id to identify this Service Profile",
      "type": "string"
    },
    "serviceProfileName": {
      "description": "Name of this service profile",
      "type": "string",
      "minLength": 1
    },
    "description": {
      "description": "Free form description field to note specifics about this
service profile",
      "type": "string",
      "maxLength": 2000
    },
    "serviceKey": {
      "description": "ServiceKey for the profile indicating what CDN service
this profile is tied to",
      "$ref": "#\/definitions\/serviceKey"
    },
    "publishedUrlPath": {
      "description": "path value to be applied to the publishedUrl",
      "type": "string",
      "pattern": "[A-Za-z0-9-.\/()%]*$",
      "minLength": 1
    },
    "sourceUrlPath": {
      "description": "path value to be applied to the sourceUrl",
      "type": "string",
      "pattern": "[A-Za-z0-9-.\/()%]*$",
      "minLength": 1
```

```
    }
  },
  "definitions": {
    "serviceKey": {
      "type": "object",
      "additionalProperties": true,
      "description": "Key that indicates what videoformat,rewriteType etc that
this Service Profiles maps to",
      "required": [
        "name"
      ],
      "properties": {
        "name": {
          "type": "string",
          "minLength": 1
        }
      }
    },
    "option": {
      "type": "object",
      "additionalProperties": false,
      "required": [
        "name"
      ],
      "properties": {
        "name": {
          "type": "string",
          "description": "This is the option name",
          "minLength": 1,
          "pattern": "^\\S+$"
        },
        "parameters": {
          "type": "array",
          "description": "This is the list of option values if any for an
option.",
          "items": {
            "type": [
              "string",
              "integer",
              "boolean"
            ],
            "minLength": 1
```

```
          }
        }
      }
    }
  }
}
```

Content DeliveryService Instance Schema

```
{
  "$schema": "http:\/\/json-schema.org\/schema#",
  "description": "DeliverySvc  Instance",
  "title": "DeliverySvcInstance",
  "additionalProperties": false,
  "required": [
    "protocolSets",
    "serviceKey"
  ],
  "type": "object",
  "properties": {
    "protocolSets": {
      "type": "array",
      "description": "Array of ProtocolSet comprising of list of possible
published protocol and sourceprotocol combination ",
      "minItems": 1,
      "maxItems": 2,
      "uniqueItems": true,
      "items": {
        "additionalProperties": false,
        "type": "object",
        "required": [
          "publishedProtocol",
          "sourceProtocol"
        ],
        "properties": {
          "publishedProtocol": {
            "description": "published Protocol ",
            "type": "string",
```

```
            "enum": [
              "http",
              "https"
            ]
          },
          "publishedPort": {
            "description": "Published  Port ",
            "type": "integer",
            "minimum": 0,
            "maximum": 65535
          },
          "sourceProtocol": {
            "description": "Source Protocol ",
            "type": "string",
            "enum": [
              "http",
              "https"
            ]
          },
          "sourcePort": {
            "description": "Source  Port ",
            "type": "integer",
            "minimum": 0,
            "maximum": 65535
          },
          "options": {
            "description": "Default values that are to be applied if the rewrite
option is not supplied in the service request",
            "minItems": 0,
            "type": "array",
            "items": [
              {
                "$ref": "#\/definitions\/option"
              }
            ]
          },
          "optionSets": {
            "description": "Names of OptionSets to be used in this service
request",
            "minItems": 0,
            "type": "array",
            "items": [
```

```
              {
                "type": "string"
              }
            ]
          }
        }
      }
    },
    "uuid": {
      "description": "Unique id to identify this DeliverySvc Instance",
      "type": "string"
    },
    "description": {
      "description": "Free form description field to note specifics about this
Service Instance",
      "type": "string",
      "maxLength": 2000
    },
    "serviceProfileName": {
      "description": "Name of the Delivery Service Profile that was used to
create this Service Instance",
      "type": "string"
    },
    "serviceProfileRelativePath": {
      "description": "Relative path of the Delivery Service Profile that was
applied to create this Service Instance ",
      "type": "string"
    },
    "publishedHostname": {
      "description": "published hostname component of URL",
      "type": "string",
      "minLength": 1,
      "maxLength": 255,
      "pattern": "^[?().\\w-]+\\.[?().\\w-]+\\.[?().\\w-]+$",
      "caseInsensitiveSearch": true
    },
    "sourceHostname": {
      "description": "source hostname component of URL",
      "type": "string",
      "minLength": 1,
      "maxLength": 255,
      "pattern": "[a-zA-z0-9.:-]*",
```

```
      "caseInsensitiveSearch": true
    },
    "publishedUrlPath": {
      "description": "user input published path value to be prepended to the
publishedUrl derived from serviceprofile",
      "type": "string",
      "pattern": "[A-Za-z0-9-.\/()%]*$",
      "minLength": 1
    },
    "sourceUrlPath": {
      "description": "user input source path value to be prepended the sourceUrl
derived from serviceprofile",
      "type": "string",
      "pattern": "^(\/[-\\w:@&?=+,.!\/~*'%$_;]*)?$",
      "minLength": 1
    },
    "serviceKey": {
      "description": "ServiceKey for the profile indicating what CDN service
this profile is tied to",
      "$ref": "#\/definitions\/serviceKey"
    }
  },
  "definitions": {
    "serviceKey": {
      "type": "object",
      "additionalProperties": true,
      "description": "Key that indicates what videoFormat,rewriteType that this
DeliveryService Instance was created for",
      "required": [
        "name"
      ],
      "properties": {
        "name": {
          "type": "string",
          "minLength": 1
        }
      }
    },
    "option": {
      "type": "object",
      "additionalProperties": false,
      "required": [
```

```
        "name"
      ],
      "properties": {
        "name": {
          "type": "string",
          "description": "This is the option name",
          "minLength": 1,
          "pattern": "^\\S+$"
        },
        "parameters": {
          "type": "array",
          "description": "This is the list of option values if any for an
option.",
          "items": {
            "type": [
              "string",
              "integer",
              "boolean"
            ]
          }
        }
      }
    }
  }
}
```

Protocol Set Schema

```
 "protocolSets": {
      "type": "array",
      "description": "Array of ProtocolSet comprising of list of possible
published protocol and sourceprotocol combination ",
      "minItems": 1,
      "maxItems": 2,
      "uniqueItems": true,
      "items": {
        "additionalProperties": false,
        "type": "object",
        "required": [
          "publishedProtocol",
```

```
          "sourceProtocol"
        ],
        "properties": {
          "publishedProtocol": {
            "description": "published Protocol ",
            "type": "string",
            "enum": [
              "http",
              "https"
            ]
          },
          "publishedPort": {
            "description": "Published  Port ",
            "type": "integer",
            "minimum": 0,
            "maximum": 65535
          },
          "sourceProtocol": {
            "description": "Source Protocol ",
            "type": "string",
            "enum": [
              "http",
              "https"
            ]
          },
          "sourcePort": {
            "description": "Source  Port ",
            "type": "integer",
            "minimum": 0,
            "maximum": 65535
          },
          "options": {
            "description": "Default values that are to be applied if the rewrite
option is not supplied in the service request",
            "minItems": 0,
            "type": "array",
            "items": [
              {
                "$ref": "#\/definitions\/option"
              }
            ]
          },
```

```
          "optionSets": {
            "description": "Names of OptionSets to be used in this service
request",
            "minItems": 0,
            "type": "array",
            "items": [
              {
                "type": "string"
              }
            ]
          }
        }
      }
    }
```