SmartPurge REST API User Guide

Content Delivery

Table of Contents

Overview

The *SmartPurge* REST API provides programmatic access to all *SmartPurge* features (most of which are also available in the *Control*). Some API features, such as unlimited callbacks, are available only with *SmartPurge Plus*, and are noted as such in *SmartPurge* Methods.

HTTP Methods

Name	Description	HTTP Method	URI Template
Submit Purge Request	Adds a new purge request to the queue	POST	/purge/v1/account/{shortname}/requests
Get Purge Request Details by ID	Get the details of a previously-submitted purge request using its ID	GET	/purge/v1/account/{shortname}/requests/{request-id}
Get Purge Requests by Shortname	Gets the details of previously-submitted purge requests	GET	/purge/v1/account/{shortname}/requests
Translate Public URL	Translates public exact URLs into corresponding origin URLs	GET	/purge/v1/account/{shortname}/translate

Authentication

The LLNW REST APIs use a combination of symmetric key cryptography and HMAC (Hashed Message Authentication Code) for message authentication and user identification. To secure all calls to the API, an HMAC digest signature is applied to every 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` – Unix time in milliseconds used to prevent 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. The 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 Sample Code below). Users may access or regenerate this key at any time by using tools in the Control under *My Settings > Edit My Profile*. `X-LLNW-Security-Token` is formed by applying MAC digest for the "data string"; i.e. `REQUEST_METHOD` + `URL` + `QUERY_STRING` (if present) + `TIMESTAMP` + `REQUEST_BODY` (if present)

Authorization

Authenticated users must have appropriate permissions assigned, on per shortname basis, in order to be authorized to manage purge requests.

Patterns

SmartPurge uses Patterns, rather than Regex, when matching multiple objects. For more information on Patterns, see *SmartPurge Pattern Details* in the *SmartPurge* section of the *Control User Guide*.

> **Note:** For "exact" matches (where `exact` is `true` in the `patterns` object of a request), the Pattern is compared as a string literal to the Published URL. Any wildcard ("*") characters in the Pattern are treated as text. For "partial" matches (where `exact` is `false` in the `patterns` object), the Pattern is compared to the Origin URL (*not* the Published URL), and wildcard characters are treated as wildcards.

Rate Limits

A maximum of 100 URLs and/or 100 Patterns may be submitted per request, and a maximum of 60 URLs and/or 60 Patterns may be submitted per minute (i.e. one URL or Pattern per second) per Account (shortname). For example: after submitting a purge request with 100 URLs, it is necessary to wait for 100 seconds before submitting the next request.

SmartPurge Methods

Request bodies should follow the schema described in the Request Schema.

Submit Purge Request

HTTP Method	`POST`
URI Template	`/purge/v1/account/{shortname}/requests`
Required URI Parameters	`shortname` - CDN Account shortname
Optional URI Parameters	None
JSON Payload	Size of JSON payload is limited to 32 kilobytes. See Purge Request Endpoint Schema (read-write, mandatory/optional attributes) for payload schema.
Description	Creates a purge request with the given JSON payload

Request Example	{
	"patterns":[
	{
	"pattern":"http://*.example.com/images/*",
	"evict":false,
	"exact":false,
	"incqs":false
	}
	],
	"email":{
	"subject":"purge results",
	"to":"user@example.com"
	},
	"callback":{
	"url":"http://test.example.com/my_callback.php"
	},
	"notes":"my first purge request"
	}

Response Example	{
	"id":"8c1a86546c3611e49c633a03000021e9",
	"states":[
	{
	"ts":1415994226253,
	"state":"queued"
	}
	],
	"username":"exampleuser",
	"shortname":"example",
	"patterns":[
	{
	"pattern":"http://*.example.com/images/*",
	"evict":false,
	"exact":false,
	"incqs":false
	}
	],

	<pre>"email":{
	"subject":"purge results",
	"to":"user@example.com"
	},
	"callback":{
	"url":"http://test.example.com/my_callback.php"
	},
	"notes":"my first purge request"
	}</pre>

Get Purge Request Details by ID

HTTP Method	`GET`
URI Template	/purge/v1/account/{shortname}/requests/{request-id}
Required URI Parameters	• `shortname` - CDN Account shortname • `request-id` - purge request ID (UUID)
Optional URI Parameters	`geostats` - return stats per-datacenter via "geostats"
Description	Returns the details of a previously submitted purge request by its id
Request Example 1	`GET https://apis.llnw.com/purge/v1/account/example/requests/8c1a86546c3611e49c633a03000021e9`

Response Example 1	<pre>{
	"id":"8c1a86546c3611e49c633a03000021e9",
	"stats":[
	{
	"pattern":0,
	"count":4,
	"size":41944984
	}
	],
	"states":[
	{
	"ts":1415994024448,
	"state":"queued"
	},
	{
	"ts":1415994024456,
	"state":"in_progress"
	},
	{
	"ts":1415994027100,
	"state":"complete"
	},
	{
	"ts":1415994036792,
	"state":"stats_avail"
	}
	],
	"patterns":[</pre>

	<pre>{
	"pattern":"http://*.example.com/images/*",
	"exact":false,
	"evict":false,
	"incqs":false
	}
	],
	"username":"exampleuser",
	"shortname":"example",
	"email":{
	"subject":"purge results",
	"to":"user@example.com"
	},
	"callback":{
	"url":"http://test.example.com/my_callback.php"
	},
	"notes":"my first purge request"
	}</pre>

Request Example 2 (geostats)	`GET https://apis.llnw.com/purge/v1/account/llnw/requests/b90d5e3c478411e59c63eaa7000005c4?geostats`

Response Example 2	<pre>{
	"id":"b90d5e3c478411e59c63eaa7000005c4",
	"geostats":{
	"dal":[
	{
	"pattern":0,
	"count":2,
	"size":512
	}
	],
	"lon":[
	{
	"pattern":0,
	"count":4,
	"size":55
	},
	{
	"pattern":3,
	"count":42,
	"size":555
	}
	]
	},
	"states":[
	{
	"ts":1415994024448,
	"state":"queued"
	},
	{
	"ts":1415994024456,</pre>

```
            "state":"in_progress"
        },
        {
            "ts":1415994027100,
            "state":"complete"
        },
        {
            "ts":1415994036792,
            "state":"stats_avail"
        }
    ],
    "patterns":[
        {
            "pattern":"http://*.example.com/images/*",
            "exact":false,
            "evict":false,
            "incqs":false
        }
    ],
    "username":"exampleuser",
    "shortname":"example",
    "email":{
        "subject":"purge results",
        "to":"user@example.com"
    },
    "callback":{
        "url":"http://test.example.com/my_callback.php"
    },
    "notes":"my first purge request"
}
```


Get Purge Requests by Shortname

HTTP Method	`GET`
URI Template	`/purge/v1/account/{shortname}/requests`
Required URI Parameters	`shortname` - CDN Account shortname
Optional URI Parameters	• `start_ts` - (milliseconds since UNIX epoch) - if given, will return purge requests submitted after this time. The default, and the oldest start time, is 90 days before the current time • `end_ts` - (milliseconds since UNIX epoch) - if given, will return purge requests submitted before this time. Cannot be more than 5 minutes later than the current time, and must be greater than the start_ts. The default is the current time. • `limit` - result set will be limited to this number of purge requests, should be in range [1, 100]. Default is 50. • `offset` - used to skip purge requests from the beginning of result set, should be in range [0, 5000]. Default is 0. • `order` - used to specify ordering of purge requests in result set (based on submission time). Can be either `desc` (descending order) or `asc` (ascending order). Default is `desc`.
Description	Returns the status of all previously-submitted purge requests for a specific shortname during the specified time frame
Request Example	`GET https://apis.llnw.com/purge/v1/account/example/requests?`

	`limit=10&offset=0`

Response Example	{
	"requests":[
	{
	"id":"8c1a86546c3611e49c633a03000021e9",
	"stats":[
	{
	"pattern":0,
	"count":4,
	"size":41944984
	}
	],
	"states":[
	{
	"ts":1415994226253,
	"state":"queued"
	},
	{
	"ts":1415994226263,
	"state":"in_progress"
	}
	],
	"username":"exampleuser",
	"shortname":"example",
	"patterns":[
	{
	"pattern":"http://*.example.com/images/*",
	"exact":false,
	"evict":false,
	"incqs":false
	}
	],
	"email":{
	"subject":"purge results",
	"to":"user@example.com"
	},
	"callback":{
	"url":"http://test.example.com/my_callback.php"
	},
	"notes":"my first purge request"
	},
	{
	"id":"59f241e46c3611e49c633a0300001fbc",
	"states":[
	{
	"ts":1415994142104,
	"state":"queued"
	},
	{
	"ts":1415994142112,

```
                    "state":"in_progress"
                },
                {
                    "ts":1415994143224,
                    "state":"complete"
                },
                {
                    "ts":1415994153449,
                    "state":"stats_avail"
                }
            ],
            "username":"exampleuser",
            "shortname":"example",
            "patterns":[
                {
                    "pattern":"http://*.example.com/css/*",
                    "exact":false,
                    "evict":true,
                    "incqs":false
                }
            ],
            "email":{
                "subject":"purge results",
                "to":"user@example.com"
            },
            "total":2
        }
    ]
}
```


Translate Public URL

HTTP Method	`GET`
URI Template	`/purge/v1/account/{shortname}/translate`
Required URI Parameters	• `shortname` - CDN Account shortname • `url` - public exact URL to translate
Optional URI Parameters	None
Description	Translates public exact URL into the corresponding origin URL
Request Example	`GET https://apis.llnw.com/purge/v1/account/example/translate?url=http://shortname.vo.llnwd.net/to/translate`

Response Example	{
	"translated":"http://origin.example.com/to/translate"
	}

Notifications

Purge Request Completion Email

If a purge request contains valid email addresses in the "email" field, upon completion an email will be sent from purge-noreply@llnw.com to those email addresses, with the following format:

```
Content purge request 6bcf2d18450e11e49c63e2a600000503 has been completed, purging
12 objects.

Thu, 25 Sep 2014 23:48:16 GMT -> request queued
Thu, 25 Sep 2014 23:48:16 GMT -> request in-progress
Thu, 25 Sep 2014 23:48:17 GMT -> request complete
Thu, 25 Sep 2014 23:48:34 GMT -> request stats available

Pattern Stats:
1: http://*/0 flags: evict; purged 1 object
2: http://*/1 flags: none; purged 1 object
3: http://*/nonexist flags: none; purged 0 objects

Tag Stats:
1: tag123 flags: none; purged 5 objects
2: tag456/* flags: evict; purged 5 objects

Request Notes:
This purge request was a test.
```

Each of the sections following the timestamps will only appear if the request contains that type of data (patterns/tags/notes). For an example of how email addresses are included in the JSON, see Submit Purge Request.

Purge Request State Callbacks

If a callback URL is provided, a GET request will be made to it when the request makes specific state transitions.

State	*SmartPurge* Callback	SmartPurge Plus Callback
`in_progress`	no	yes
`complete`	no	yes
`stats_avail`	yes	yes

The callback will include query parameters:

- `purge_request_id` - the UUID of the purge request without dashes
- `purge_request_state` - the new external state string as listed in the request endpoint schema

For example: `http://test.example.com/my_callback.php?purge_request_id=7b4cb3865a0e11e49c63e2a600000354&purge_request_state=stats_avail`

See the Sample Code for examples of how to provide the callback URL.

API Activation

Understanding Your Welcome Letter

When your *SmartPurge* API service is ready for use, each of the Technical Contacts for your Limelight Account will receive an email Welcome Letter from the Limelight NOC (Network Operations Center) at support@llnw.com.

The letter will include any information you need to enter to activate and configure the service, and also provides contact information for the Limelight NOC, additional information on troubleshooting and escalation, and background information on maintenance notifications.

Test and Tune

Using Test Content

Limelight recommends you use test-only (non-production) content to test the *SmartPurge* API.

No Endpoint URI Response

You can use your browser to view the response from the endpoint without any query string terms. The response will be an HTTP 4xx error.

Invalid Credentials

If the credentials you provide during authentication are incorrect, the response will be an HTTP 401 “Unauthorized” error. If this occurs, please verify you are using the credentials provided in your Welcome Letter.

Credentials Lacking Proper Permissions

If the credentials you provide during authentication lack the proper permissions for the specified shortname, the response will be an HTTP 403 “Forbidden” error. If this occurs, please verify you are using the credentials provided in your Welcome Letter.

Incorrect Shortname

If you use an incorrect shortname (one which is invalid or not authorized for your credentials) in a method, the response will be an HTTP 403 “Forbidden” error.

Invalid Request ID

If you use an invalid Request ID in a method, the response will be an HTTP 400 “Bad Request” error with message code 1011.

Too Many Requests

If an excessive number of purge requests are submitted, the request will be rejected with a 429 HTTP Code (Too Many Requests).

Manage the *SmartPurge* API

Best Practices

Objects are normally updated in or removed from the Limelight CDN cache during “freshness checks” with your origin. For a given object, a freshness check is initiated when a request has been made for the object, and EdgePrism determines that the object’s TTL has expired. Since TTL directly controls caching, setting object TTLs is the best and most efficient way to manage cached content.

For example, a news site may need to provide rapid updates to a breaking video story. The video can be updated in cache as quickly as desired by assigning it a low TTL value via a cache control response header. There is no need to directly “purge” the video from cache.

However, there are special cases where content needs to be flagged for update on the next user request, or even proactively removed from cache as soon as possible. Examples of when this might be required include:

- You need to update an old file with a new file of the same filename
- You discover that some of your cached content is infringing on another’s copyright, and need to delete the content from cache as soon as possible
- You lose a contract with a content provider, and obligated to delete the provider’s content from your cache as soon as possible
- Purging is also appropriate during a full website update, when you need to quickly update many related website objects (images, text, video, etc.) at the same time.

CAUTION: To avoid a “too many requests” error when submitting a purge request, Limelight recommends you provide one or more patterns, rather than a fully-qualified URL for each object.

Cache-Tag Response Headers

Example Cache-Tag: *foo,bar,baz*

You can use tags with SmartPurge. These content tags get associated with an object when it is first cached by the CDN; existing objects do not get tags associated on a refresh check. Once the tag is associated with an object, you can use the SmartPurge user interface or API to purge using individual content tag values.

Cache-Tags in SmartPurge should use this format:

- max length 64 characters (for the entire header value, not the individual tags
- designated separator is a , (comma)
- ASCII-printable characters (character code 32-126) with these exceptions
  - whitespace
  - designated separator (comma)
  - delete

Track Usage

You can use the *SmartPurge* API methods Get Purge Request Details by ID and Get Purge Requests by Shortname to track usage and purge request status.

Appendix

Request Endpoint Schema

Following is a single purge request object schema:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><code>id</code></td><td>string
read-only</td><td>Purge request identifier (UUID), e.g. 8c1a86546c3611e49c633a03000021e9</td></tr>
<tr><td><code>states</code></td><td>array of objects
read-only</td><td>Transition times for <code>in_progress</code> and <code>complete</code> are only available for SmartPurge Plus customers. This value is an array of objects with the following schema:
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><code>ts</code></td><td>integer</td><td>Time (milliseconds since UNIX epoch) when purge request reached this state</td></tr>
<tr><td><code>state</code></td><td>string</td><td>Purge request state, possible values:
<code>queued</code> - purge request is queued for processing
<code>in_progress</code> - purge request is being processed (SmartPurge Plus only)
<code>complete</code> - purge request processing is complete (SmartPurge Plus only)
<code>stats_avail</code> - purge request processing statistics is available</td></tr>
</table></td></tr>
<tr><td><code>username</code></td><td>string
read-only</td><td>Submitter/owner of this purge request</td></tr>
<tr><td><code>shortname</code></td><td>string
read-only</td><td>CDN Account shortname</td></tr>
<tr><td><code>patterns</code></td><td>array of objects
read-write
(optional)</td><td>List of patterns and/or exact URLs to purge, an array of objects with the following schema:
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><code>pattern</code></td><td>string
(mandatory)</td><td>pattern or exact URL to purge. 4096 characters maximum</td></tr>
<tr><td><code>evict</code></td><td>boolean
(mandatory)</td><td>if true, matching cache objects are evicted (deleted), otherwise invalidated</td></tr>
<tr><td><code>exact</code></td><td>boolean
(mandatory)</td><td>if true, pattern is treated as an exact public URL, otherwise pattern must be based on origin URL</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td><code>incqs</code></td><td>boolean
(mandatory)</td><td>if true, pattern is allowed to match query string part of URL, otherwise query string is stripped before matching</td></tr>
</table>

Note: Purge request must have at least one pattern or content tag. Patterns and tags can be combined in one request.
</td></tr>
<tr><td><code>tags</code></td><td>array of objects
read-write
(optional)</td><td>List of content tags to purge, an array of objects with the following schema:
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><code>tag</code></td><td>string
(mandatory)</td><td>content tag to purge. 256 characters maximum</td></tr>
<tr><td><code>evict</code></td><td>boolean
(mandatory)</td><td>if true, matching cache objects are evicted (deleted), otherwise invalidated</td></tr>
</table>

Note: Purge request must have at least one pattern or content tag. Patterns and tags can be combined in one request.
</td></tr>
<tr><td><code>email</code></td><td>object
read-write
(optional)</td><td>Email address(es) to receive purge request completion notification email, an object with the following schema:
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><code>subject</code></td><td>string
(optional)</td><td>subject of notification email. 128 characters maximum</td></tr>
<tr><td><code>to</code></td><td>string
(mandatory)</td><td>notification email recipient, comma should be used to separate multiple recipients. 256 characters maximum</td></tr>
<tr><td><code>cc</code></td><td>string
(optional)</td><td>notification email carbon copy, comma should be used to separate multiple recipients. 256 characters maximum</td></tr>
<tr><td><code>bcc</code></td><td>string
(optional)</td><td>notification email blind carbon copy, comma should be used to separate multiple recipients. 256 characters maximum</td></tr>
</table>
</td></tr>
<tr><td><code>callback</code></td><td>object
read-write
(optional)</td><td>HTTP(S) callback URL for purge request state transition notifications, an object with the following schema:
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><code>url</code></td><td>string
(mandatory)</td><td>URL to be used for state transition notifications</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td></td><td><table><tr><td></td><td></td><td>Userinfo, query string, and fragment must be empty. 512 characters maximum.</td></tr></table></td></tr>
<tr><td><code>stats</code></td><td>array of objects
read-only</td><td>Purge request pattern and/or content statistics, an array of objects with the following schema:<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td><code>pattern</code> or <code>tag</code></td><td>integer</td><td>Index of pattern or content tag in patterns or tags purge request array</td></tr><tr><td><code>count</code></td><td>integer</td><td>Number of cache objects purged by this pattern or content tag</td></tr><tr><td><code>size</code></td><td>integer</td><td>Total size of cache objects (in bytes) purged by this pattern or content tag</td></tr></table></td></tr>
<tr><td>geostats</td><td>object read-only</td><td>Statistics for the request broken down by datacenter. Replaces the "stats" entry when the request query contains "geostats". The value of each member is an array of pattern and/or content tag stats for this member. Single-request only.</td></tr>
<tr><td><code>aborted</code></td><td>boolean
read-only</td><td>Indicates that purge request was aborted due to internal limitations.</td></tr>
<tr><td><code>completion</code></td><td>double
read-only</td><td>Purge request completion percentage. Present only for SmartPurge Plus customers if request in the <code>in_progress</code> state. Single-request only.</td></tr>
<tr><td><code>notes</code></td><td>string read-write (optional)</td><td>Purge request notes. Arbitrary text describing this purge request. 512 characters maximum.</td></tr>
<tr><td>dry-run</td><td>boolean read-write (optional)</td><td>If true, the request is executed without deleting or invalidating any cache objects. Resulting statistics indicate the number of cache objects that would have been affected if the request were not submitted with dry-run=true.</td></tr>
</table>

Multiple Purge Request Response Schema

When multiple purge request objects are returned, they are encoded as an object of the following schema:

Name	Type	Description
`requests`	array of objects	Purge requests, an array of objects of the schema given above, except those labeled “single-request only”, which are not provided.
`total`	integer	Estimated number of total purge request objects satisfying this API request This number can be used to estimate number of pages if pagination is desired. As this number is an estimation, it can be larger

		than the actual number of purge request objects.
`more`	boolean	If true, indicates that there are more than total purge request objects satisfying this API request, but only total number of them are available due to system limits (current limit for the total number of purge requests in a result set is 5000)

Error Response Schema

Following is an error response object schema:

Name	Type	Description
`errors`	array of objects	API request processing errors, an array of objects with the following schema: Name — Type — Description `message` — string — Error message `code` — integer — Error code `description` — string — Detailed error description `source` — string — Source of the error

Following is an example of an error response object:

Error Response Example

```
{
    "errors":[
        {
            "message":"missing url",
            "code":1019,
            "description":"missing required query parameter: url",
            "source":"query"
        }
    ]
}
```

Error Response Descriptions

The following errors can be returned by *SmartPurge* (description and source may vary for different errors):

Message	Code	Description	Source
`missing required property`	1001	The required property incqs is missing.	patterns[1]
`no extra properties allowed`	1003	The unknown property size is not allowed.	patterns[2].size

Message	Code	Description	Source
`invalid type`	1004	The property incqs has an invalid type. The required type is boolean.	patterns[3].incqs
`invalid size`	1005	The size of the property patterns must be in the range [1 - 100].	patterns
`invalid length`	1006	The length of the property subject must be in the range [1 - 128].	email.subject
`invalid pattern`	1007	The pattern foo* bar* is invalid.	patterns[1].pattern
`unconfigured URL`	1008	The exact public URL example.com does not match any Published Host in the configuration for the specified short-name/account. **Note:** The exact property must be false to use pattern globbing via *.	patterns[5].pattern
`malformed JSON body`	1009	The JSON request body is malformed.	request body
`invalid timestamp`	1010	The X-LLNW-Security-Timestamp request header value foo is invalid. The required type is integer.	security timestamp
`invalid request id`	1011	The SmartPurge request id foo not a valid UUID.	purge request id
`invalid offset`	1012	The query string term offset value -1 is invalid. It must be an integer in range [0 - 5000].	offset query parameter
`invalid limit`	1013	The query string term limit value -1 is invalid. It must be an integer in range [1 - 100].	limit query parameter
`invalid start_ts`	1014	The query string term start_ts value -1 is invalid. It must be a number of mil-	start_ ts query parameter

Message	Code	Description	Source
		liseconds not more than 90 days before the time of submission.	
`invalid end_ts`	1015	The query string term end_ts value -1 is invalid. It must be a number of milliseconds not more than 90 days before the time of submission and greater than the start_ts value.	end_ts query parameter
`invalid timestamp range`	1016	The value of the start_ts query string term must be less than the value of the end_ts term.	query string
`invalid order`	1017	The query string term order value foo is invalid. It must be either asc or desc.	order query parameter
`invalid filter parameter`	1018	The filter parameter delay=foo is invalid. It must be an integer.	corrupt filter parameter
`missing URL`	1019	The required query string term url is missing.	query string
`invalid query string`	1020	The query string =1 is malformed.	query string
`queued patterns limit is reached`	1021	You have reached the maximum number of queued patterns (1000). Please resubmit your request after some of your patterns have been processed.	system limits
`patterns per minute limit is reached`	1022	You have reached the maximum number of submitted patterns per minute (60). Please wait 60 seconds before resubmitting your request.	system limits
`invalid URL`	1023	The URL 'http://foo/\n' in your request is invalid.	url query parameter
`user authentication failed`	1024	The user information in your X-LLNW-Security-* headers could not be	user authentication

Message	Code	Description	Source
		authenticated.	
`user authorization failed`	1025	The user specified in the X-LLNW-Security-Principal request header is not authorized to perform this operation.	user authorization
`invalid token`	1026	The value in the X-LLNW-Security-Token request header is invalid.	security token
`invalid node id`	1027	The node id foo is invalid.	node id
`invalid email`	1028	The property email.to contains the invalid email address foo.	email.to
`invalid callback URL`	1029	The property callback.url contains the invalid URL foo.	callback.url
`not in range`	1030	The property foo must be an integer in range [0-4294967295].	pending_patterns
`unconfigured URL`	1031	The exact public URL example.com does not match any Published Host in the configuration. Please correct the values of the URL or exact flag.	url query parameter
`URL metadata unavailable`	1032	The metadata for the exact public URL example.com is temporarily unavailable. Please resubmit your request later. If the problem persists, please contact Customer Service.	url query parameter
`translation unavailable`	1033	The exact public URL value example.com in your request could not be translated. Please resubmit your request later. If the problem persists, please contact Customer Service.	url query parameter
`internal translation exceeds limits`	1034	The exact public URLs in your request could not be purged all at once. Please	system limits

Message	Code	Description	Source
		resubmit your request with fewer exact URLs.	
`cache configuration error`	1035	This purge request cannot be completed. An exact public URL in this request matches a misconfigured Published Host. Please contact Customer Service.	cache configuration
`translation unavailable`	1036	This service is temporarily unavailable. Please resubmit your request later. If the problem persists, please contact Customer Service.	system error
`account misconfiguration`	1037	An account mis-configuration prevents the request for pattern 'http://example.com' from proceeding. Please contact Customer service to resolve this issue.	patterns[1].pattern
`feature unavailable`	1039	The feature 'featurename' is not available at this time.	featurename
`invalid tag`	1040	The tag 'foo bar' is invalid.	tags[1].tag
`request is too big`	1041	The combined size of patterns and tags arrays must not exceed 100.	patterns and tags
`request is empty`	1042	Please specify patterns and/or tags in your purge request.	patterns and tags

HTTP Status Codes

HTTP Status Code	Description	Has Body?
`200`	GET, PUT or OPTIONS request has been successfully processed	Yes
`201`	POST request successfully created a new entity	Yes
`204`	DELETE request successfully deleted an entity	No
`400`	Request could not be processed	Yes
`401`	Request is unauthorized	Yes

HTTP Status Code	Description	Has Body?
403	Request is forbidden	Yes
404	Requested resource is not available	No
405	HTTP Method not allowed	No
422	Unprocessable Entity	No
413	Request Entity Too Large	No
429	Too Many Requests	Yes
500	Internal server error	No
503	Service is unavailable	No

Sample Code

Python Sample Code

```
                import urllib2
import time
import hmac
import hashlib
try: import simplejson as json
except ImportError: import json

class PurgeApi:
    def __init__(self, apiShortName, apiUser, apiKey):
        self.baseUrl = "https://apis.llnw.com/purge/v1/account/" + apiShortName +
"/requests"
        self.apiUser = apiUser
        self.apiKey = apiKey

    def getStatusByShortName(self, start_ts=None, end_ts=None, limit=50, offset=0):
        if start_ts is None:
            start_ts = int(round(time.time()*1000)) - 2592000000
        if end_ts is None:
            end_ts = int(round(time.time()*1000))
        url = self.baseUrl + "?start_ts=" + str(start_ts)
        url += "&end_ts=" + str(end_ts)
        url += "&limit=" + str(limit)
        url += "&offset=" + str(offset)
        return self.get(url)

    def getStatusById(self, apiRequestId):
        url = self.baseUrl + "/" + apiRequestId
```

```
        return self.get(url)

    def createPurgeRequest(self, body):
        return self.post(self.baseUrl, json.dumps(body))

    def get(self, url):
        return self.execute("GET", url, "")

    def post(self, url, body):
        return self.execute("POST", url, body)

    def execute(self, method, url, body):
        opener = self.__getOpener()
        request = self.__buildRequest(method, url, body)
        try:
            connection = opener.open(request)
            response = connection.read()
        except urllib2.URLError, e:
            # handle error response here
            return ""
        else:
            return json.loads(response)

    def __getOpener(self):
        return urllib2.build_opener(urllib2.HTTPHandler)

    def __generateHMAC(self, data):
        return hmac.new(self.apiKey.decode('hex'), msg=data,
digestmod=hashlib.sha256).hexdigest()

    def __buildRequest(self, method, url, body):
        request = urllib2.Request(url, data=body)
        request.get_method = lambda:method
        auth_url = url.partition("?")[0]
        qs = url.partition("?")[2]
        request.add_header('Content-Type','application/json')
        request.add_header('X-LLNW-Security-Principal', self.apiUser)
        timestamp = str(int(round(time.time()*1000)))
        request.add_header('X-LLNW-Security-Timestamp',timestamp)
        request.add_header('X-LLNW-Security-Token', self.__generateHMAC(method + auth_
url + qs + timestamp + body))
        return request

def main():
    apiKey = "your_key_goes_here"
    apiUser = "your_user_goes_here"
```

```
    apiShortName = "your_shortname_goes_here"
    apiRequestId = "your_request_id_goes_here"

    api = PurgeApi(apiShortName, apiUser, apiKey)

    result = api.getStatusByShortName()
    print json.dumps(result, sort_keys=True, indent=4)

    result = api.getStatusById(apiRequestId)
    print json.dumps(result, sort_keys=True, indent=4)

    patterns = [
        {"pattern" : "http://*.net", "evict" : False, "exact" : False, "incqs" :
False},
        {"pattern" : "http://test.url", "evict" : False, "exact" : False, "incqs" :
False}
    ]
    email = {
        "subject" : "some subject",
        "to" : "email_to@test.com",
        "cc" : "email_cc@test.com",
        "bcc" : "email_bcc@test.com"
    }
    callback ={"url" : "http://some.callback.url"}
    requestBody = {
        "patterns" : patterns,
        "email" : email,
        "callback" : callback
    }

    result = api.createPurgeRequest(requestBody)
    print json.dumps(result, sort_keys=True, indent=4)

if __name__ == "__main__":
    main()
```

Java Sample Code

```
import java.io.IOException;
import java.io.InputStream;
import java.io.StringWriter;
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;
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.HttpMethodBase;
import org.apache.commons.httpclient.HttpStatus;
import org.apache.commons.httpclient.methods.GetMethod;
import org.apache.commons.httpclient.methods.PostMethod;
import org.apache.commons.io.IOUtils;

public class SimpleClient {
    private static final int DEFAULT_OFFSET = 0;
    private static final int DEFAULT_LIMIT = 50;
    private static final String algorithm = "HmacSHA256";
    private String apiKey;
    private String apiUser;
    private String baseUrl;

    public SimpleClient(String apiShortName, String apiUser, String apiKey) {
        this.baseUrl = "https://apis.llnw.com/purge/v1/account/" + apiShortName +
"/requests";
        this.apiUser = apiUser;
        this.apiKey = apiKey;
    }

    public String getStatusByShortName() {
        long now = System.currentTimeMillis();
        return getStatusByShortName(now - 2592000000L, now, DEFAULT_LIMIT, DEFAULT_
OFFSET);
    }

    public String getStatusByShortName(long start_ts, long end_ts, int limit, int
offset) {
        String url = baseUrl + "?start_ts=" + start_ts +
            "&end_ts=" + end_ts +
            "&limit=" + limit +
            "&offset=" + offset;
        return get(url);
    }

    public String getStatusById(String apiRequestId) {
        String url = baseUrl + "/" + apiRequestId;
        return get(url);
    }

    public String createPurgeRequest(String body) {
        return post(baseUrl, body);
    }
```

```
    private String get(String url) {
        GetMethod method = new GetMethod(url);
        int rc = execute(method, "GET", url, "");

        if (HttpStatus.SC_OK != rc) {
            /* Handle errors with the API communication */
        }
        return getBodyAsString(method);
    }

    private String post(String url, String body) {
        PostMethod method = new PostMethod(url);
        method.setRequestBody(body);
        int rc = execute(method, "POST", url, body);

        if (HttpStatus.SC_CREATED != rc) {
            /* Handle errors with the API communication */
        }
        return getBodyAsString(method);
    }

    private int execute(HttpMethodBase methodBase, String method, String url, String
body) {
        try {
            String timestamp = Long.toString(System.currentTimeMillis());
            String auth_url = url;
            String qs = "";
            int idx;
            if ((idx = url.indexOf('?')) > 0) {
                auth_url = url.substring(0, idx);
                qs = url.substring(idx + 1);
            }
            String digest = new String(Hex.encodeHex(generateHMAC(
                (method + auth_url + qs + timestamp + body).getBytes("UTF-8"))));
            methodBase.addRequestHeader("Content-Type", "application/json");
            methodBase.addRequestHeader("X-LLNW-Security-Token", digest);
            methodBase.addRequestHeader("X-LLNW-Security-Principal", apiUser);
            methodBase.addRequestHeader("X-LLNW-Security-Timestamp", timestamp);

            return (new HttpClient()).executeMethod(methodBase);
        } catch (Exception e) {
            /* Handle error here*/
            return -1;
        }
    }
```

```
    private String getBodyAsString(HttpMethodBase methodBase) {
        try {
            InputStream is = methodBase.getResponseBodyAsStream();
            StringWriter writer = new StringWriter();
            IOUtils.copy(is, writer);
            String theString = writer.toString();
            return theString;
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        } finally {
            methodBase.releaseConnection();
        }
    }

    private byte[] generateHMAC(byte[] data) throws DecoderException,
InvalidKeyException, NoSuchAlgorithmException {
        byte[] sharedKey = Hex.decodeHex(apiKey.toCharArray());
        SecretKeySpec keySpec = new SecretKeySpec(sharedKey, algorithm);
        Mac mac = Mac.getInstance(algorithm);
        mac.reset();
        mac.init(keySpec);
        return mac.doFinal(data);
    }

    public static void main(String[] args) {
        String apiKey = "your_key_goes_here";
        String apiUser = "your_user_goes_here";
        String apiShortName = "your_shortname_goes_here";
        String apiRequestId = "your_request_id_goes_here";

        String requestBody =
            "{\"patterns\" : [" +
                "{\"pattern\" : \"http://*.net\", \"evict\" : false, \"exact\" : false,
\"incqs\" : false}," +
                "{\"pattern\" : \"http://test.url\", \"evict\" : false, \"exact\" :
false, \"incqs\" : false}" +
            "]," +
            "\"email\" : {" +
                "\"subject\" : \"some subject\"," +
                "\"to\" : \"email_to@test.com\"," +
                "\"cc\" : \"email_cc@test.com\"," +
                "\"bcc\" : \"email_bcc@test.com\"" +
            "}," +
            "\"callback\" : {" +
```

```
                "\"url\" : \"http://some.callback.url\"" +
            "}}";

        SimpleClient sc = new SimpleClient(apiShortName, apiUser, apiKey);
        System.out.println(sc.getStatusByShortName());
        System.out.println(sc.getStatusById(apiRequestId));
        System.out.println(sc.createPurgeRequest(requestBody));
    }
}
```