Lifecycle Manager Import API

Lifecycle Manager

Import API
Version 7.0
January, 2016

Copyright

Trademarks

Akana, Inc.

Akana, Inc.
12100 Wilshire Blvd, Suite 1800
Los Angeles, CA 90025
(866) SOA-9876
www.akana.com
info@akana.com

Disclaimer

Contents

Chapter 1 | Overview

The ImportUtils API is a REST (HTTP) interface for programmatically invoking and monitoring Asset import jobs utilizing the DelimitedFileAssetImporter.

The API is enabled by declaring it as a function in the LPC:

```
<function authentication="BASIC" class="ImportUtils" name="importutils" />
```

The “authentication” attribute can be “NONE” for no authentication, “BASIC” for HTTP basic authentication.

The “name” attribute determines the path for the HTTP calls. Assuming this attribute is set to “importutils” The base URI for the operations described in this document is:

http://<host:port>/lm/custom/rest/<library name>/importutils.

Chapter 2 | Operations

Import Assets

This function is the equivalent of running the DelimitedFileAssetImporter from the LM UI. It responds with a "run id" that can be used to monitor the progress of the import.

HTTP Operation:

POST

Path:

…/import

Request Content Type

JSON (if import file specified as URL)
multipart/mixed (if import file attached)

Parameters

None

Request Details

The request should consist of a JSON part specifying the import parameters and import document. If the import document is not specified as a URL then a File part must be provided on the request with a name matching that specified in the JSON Document element.

Here's an example JSON import request for a URL type import document showing all possible properties:

```
{
        "user": "support",
        "concurrentThread": 1,
        "loadFactor": 10,
        "attributeDelimiter": "|",
        "attributeValueDelimiter": "^",
        "threadSleep": 250,
        "replace": true,
        "emailResults": false,
        "ignoreAssetId": false,
        "bypassGovernance": true,
        "validate": true,
        "submissionNote":"A submission note",
        "importFile": {
                "url": "http://acme.com/imports/contents_LibraryDemo.zip"
        },
        "submit": true
```

```
}
```

Note that all properties with the exception of “importFile” are optional.

The properties "concurrentThread","loadFactor", "attributeDelimiter", "attributeValueDelimiter",”threadSleep", "emailResults", "ignoreAssetId", and "bypassGovernance" correspond to configuration properties for the DelimitedFileAssetImporter and are described in the Importer section of the Library Configuration Guide.

The properties “importFile”, “validate”, “replace”, and “submissionNote” correspond to input parameters for the DelimitedFileAssetImporter and are also described in the Library Configuration Guide.

The “user” property is optional and allows a user to be specified for use in creating and submitting imported assets. The specified user must have the ACE role. If not specified and BASIC mode is used, the specified basic authentication user will be used. Otherwise, the system user will be used.

The “importFile” property may be of the form of a URL:

```
 "importFile": {
          "url": http://acme.com/imports/contents_LibraryDemo.zip
 }
```

Or an attached file:

```
"importFile": {
     "name": "contents_LibraryDemo.zip"
}
```

In the latter case, the import request must be of multipart/mixed with the import file attached as a “file” part with a name matching that used in the “name” property.

Response Codes:

- **200** – Import file submitted for processing (note this does not indicate the success of the import itself).
- **400** – The request was invalid in some way. The response body will be text containing details about the error.
- **401 –** Occurs if BASIC authentication is used and the basic authentication credentials are missing or incorrect. May also occur if a specified user does not have ACE role.
- **500** – Import could not be processed due to an unexpected error within the server. The response body will be text containing details about the error.

Response

The response body will be of type JSON with the following properties:

- “run-id”
  The id of the import run. This id can be used to check the status of the import.
- “parsingErrorsExist”
  Indicates that some rows of the import were not successfully parsed.

- "assetImportMessages"
  An array of import result properties for each row that failed to parse correctly. These properties will contain the following.
  - "valid"
    Indicates that the row was valid. This will be false in this scenario.
  - "assetName"
    Name of the asset to import.
  - "assetVersion"
    Version of the asset to import.
  - "assetId"
    Id of the asset to import.
  - "assetDescription"
    Description of the asset to import.
  - resultMessage
    Parsing error message

Example Response:

```
{
        "parsingErrorsExist": false,
        "runid": "import_2016-01-22_15-35-07.484",
        "assetImportMessages": []
}
```

Clear Assets

This method is used to remove all assets and asset requests from a library. Since there is no way to undo this operation is should be used with caution. If run with BASIC authentication mode, the specified user in the basic authentication credentials must have the Usage Controller role for the Enterprise Group. If run with authentication mode of NONE the system user will be used.

HTTP Operation:

POST

Path

…/clearassets

Request Content Type

N/A

Parameters

- **clearevents**
  Passing "true" for the clearevents indicates that all pending events in the event queue should be removed also. This can be useful for cases where there is still pending asset activity at the time of the clearassets call. The default for clearevents is "false".

Request Details
No request body is expected

Response Codes:
- **200** – Assets, requests and optionally events were successfully removed
- **401** – Occurs if BASIC authentication is used and the basic authentication credentials are missing or incorrect. May also occur if the specified basica authentication user does not have Usage Controller authority for the Enterprise Group.
- **500** – Operation could not be performed due to an unexpected error within the server. The response body will be text containing details about the error.

Response:
N/A

Get Import Status
This method is used to check the status of an import job.

HTTP Operation:
GET

Path
…/getimportstatus

Request Content Type
N/A

Parameters
- **runid**– run id of the import job. This is the "runid" property returned from the import operation.

Request Details
No request body is expected.

Response Codes:
- **200** – Status successfully retrieved
- **500** – Status could not be retrieved due to an unexpected error within the server. The response body will be text containing details about the error.

Response:
The response body will be of type JSON with the following property:

- "unpublished-assets"
  This is the count of assets from the specified import that are not yet published. If the total asset count for an import run are not yet calculated, a value of -1 will be returned. A value of 0 indicates the import is completed. Since not all assets in an import may successfully publish, the returned value for this call may never reach 0. It is expected that clients will poll on this method, looking for either 0 or a positive result that does not change over a period of time to conclude that an import has completed.

Get Library Status

This method is used to retrieve statistical information about the activity in a library.

HTTP Operation:

GET

Path

…/getlibrarystatus

Request Content Type

N/A

Request Details

No request body is expected.

Response Codes:

- **200** – Library statistics successfully retrieved
- **500** – Library statistics could not be retrieved due an unexpected error within the server. The response body will be text containing details about the error.

Response:

The response body will be of type JSON with the following properties:

- "catalogAssets"
  The number of assets in the catalog (this is the total count of assets in a library).
- "publishedAssets"
  The number of published assets in the library.
- "projects"
  The number of projects in the library.
- "users"
  The number of users in the library.
- "activeRequests"
  The number of requests in the library that have a non-completed status.
- "pendingEvents"
  The number of events in the event queue.

Example Response:


```
{
	"catalogAssets": 11,
	"publishedAssets": 11,
	"projects": 3,
	"users": 16,
	"activeRequests": 0,
	"pendingEvents": 0
}
```