Message Studio REST API

REST API Guide

9.5

Selligent

Contents

Introduction

The Message Studio application supports a Representational State Transfer (REST) application programming interface. REST is beneficial to use instead of a SOAP API because REST is focused on accessing named resources through a single, consistent interface.

The SOAP interface, by contrast, exposes named operations used to implement some business logic through different interfaces. Also, unlike a SOAP API, REST provides true web services based on Universal Resource Indicators (URIs) and HTTP.

The following sections provide more details on the Message Studio REST API.

Topics:

- **REST API Framework on page 5**
- **REST API Authentication on page 7**
- **Mailing API Operations on page 10**
- **Data Source API Operations on page 15**
- **Program API Operations on page 21**

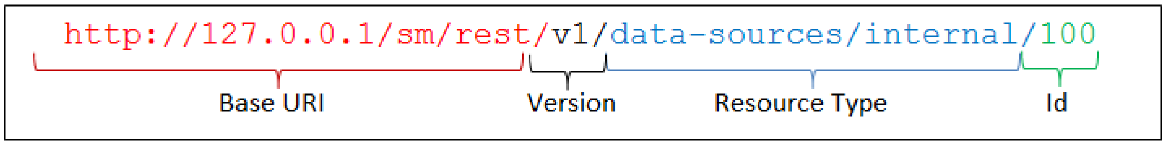
REST API Framework

The section describes the framework and supporting structures for the Message Studio REST API. This framework is built primarily using the following libraries:

- Spring is used for the MVC services
- Orika is used for mapping API beans to back-end entity objects
- Jackson is used as the JSON converter

URI Structure

The URI indicates a collection of resources and, by appending an ID to the URI, one can get to a specific resource. A sample URI and its components are shown below.



You can also add a term after the ID to indicate an operation on a particular resource. In the above example, adding "import" could mean that the API supports data source import. One important distinction to make is the difference between the URI and query parameters.

To determine if the term must be put into the URI or the query parameter, see if the term under consideration indicates modification of the output of another "action" or whether it indicates an action itself. For example, if a data import involved some kind of character encoding, the encoding may be provided as part of the query parameter.

HTTP Methods

The HTTP methods describe the type of action to perform on a resource, as explained below:

- **GET**—This method indicates that a resource is being fetched. All GET operations are read-only. Any changes to the back-end state of a resource must be initiated by the client via one of the other methods.

- **POST**—This method is used to either create a new resource or create a new dependency for the resource. For example, POST can also be used for uploading a file, which creates a new dependency for the data source. However, POST should never be used to update a resource. Instead, use a PUT method to update the resource.
- **PUT**—This method is used whenever a resource or a resource dependency must be updated. PUT may also be used when a data source needs to be posted to the back end in a way that it doesn’t change the back-end state.
- **DELETE**—Removes a resource. A message indicating success is returned.

CRUD Operations

Each asset in Message Studio requires CRUD (Create, Read, Update and Delete) operations. To maintain uniformity within the software, we define common conventions that indicate the URI, method and other required parameters for CRUD operations. This is done in such a way that every module implements the same interface and will automatically have the URI/Method pattern assigned to the CRUD APIs.

For details on how the CRUD APIs look, please refer to the API operations in the following sections. In addition to CRUD, the various assets also have other common functions such as copy and list.

Versioning & Response Format

The Message Studio REST API puts the version in the URI to avoid breaking existing applications when the system is upgraded to a new version.

The format is determined by the HTTP Accept header. A lack of an Accept header or */* as the mime will result in the response type defaulting to JSON.

REST API Authentication

The authentication of the Message Studio REST API is based on an API key that must be generated by the user.

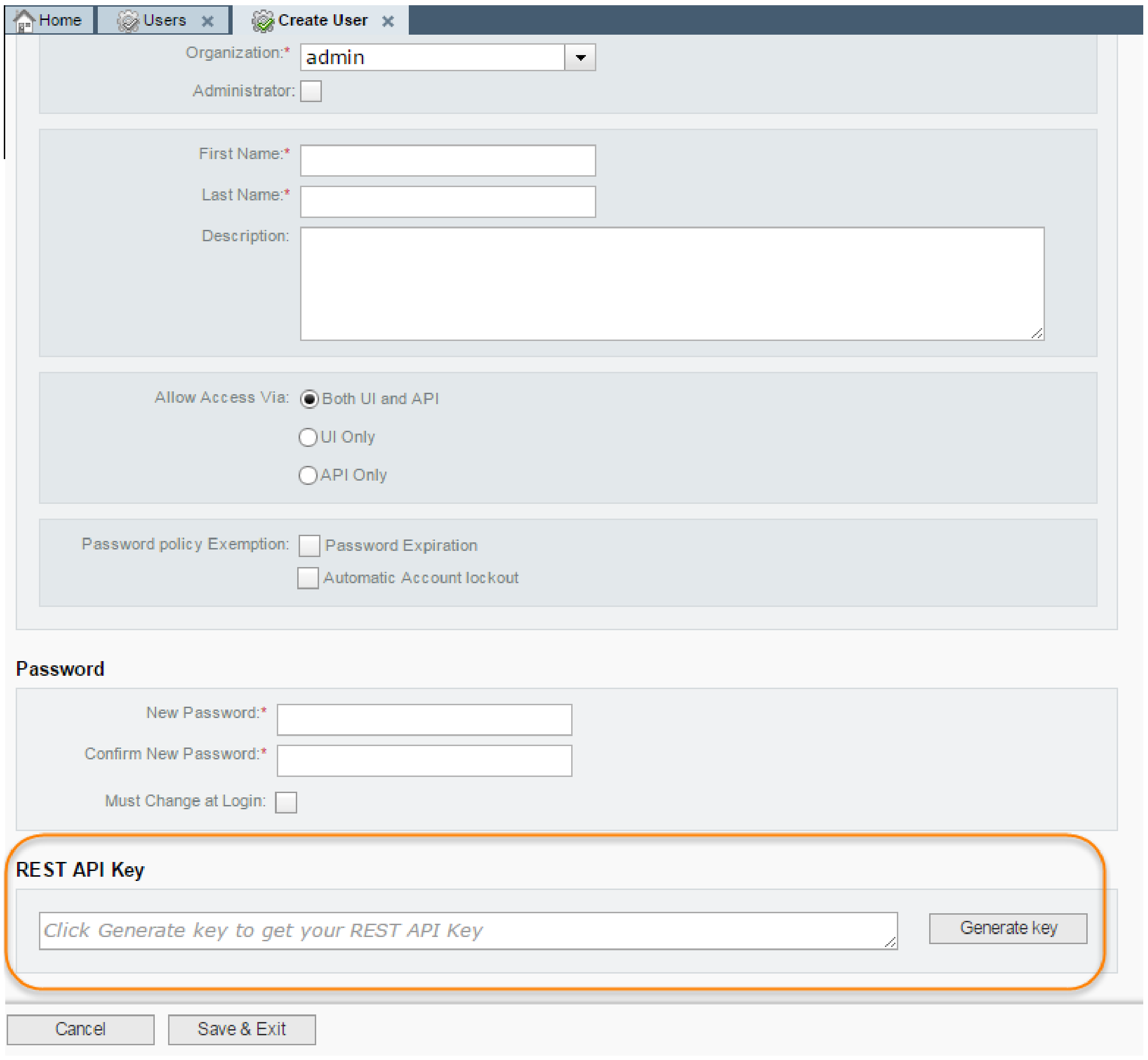
Important: Only an **Administrator** user can generate REST API keys. See **Settings** > **Users** in Message Studio for user privileges.

The process of creating the key is a two-step process:

1. Generate the REST API key in Message Studio UI for each user who will be using the REST API.
2. Use the REST API key as a HTTP header in the REST API requests.

To Generate a REST API key:

1. Log in to the Message Studio application as an Administrator user.
2. Select **Settings** > **Organization** > **Users** and either **Create** or **Edit**.
3. Under the **REST API Key** option, select **Generate key** (or **Regenerate key**, if editing an existing user).



Organization:*	admin
Administrator:	☐
First Name:*	
Last Name:*	
Description:	
Allow Access Via:	[✓ Both UI and API] UI Only API Only
Password policy Exemption:	☐ Password Expiration ☐ Automatic Account lockout

Password

New Password:*	
Confirm New Password:*	
Must Change at Login:	☐

REST API Key

Click Generate key to get your REST API Key

Note: Once a key is generated and saved, it cannot be retrieved. The Administrator must communicate it to the intended user and the user must copy it for reference. Editing the User account does not show the key again. The key can be regenerated if it is ever lost. If the key must be regenerated, you must also update the REST API injector with the new key.

Using the API Key:

The REST API key should be supplied using an HTTP Authorization header. Along with the Authorization header, a custom HTTP header X-Organization must be provided in the REST API request.

The HTTP header should be set to the fully qualified name of the Organization to which the intended operation or the asset belongs.

Thus, a sample REST request would look like this:

```
curl -H 'Authorization:
00016fed57a22738de5d187dd1fdfa61838eae79e2b0c9fb279edb3a5623b29606ac’ -H 'X-
Organization: <organization>' <REST API URL> <Request Data>
```

Mailing API Operations

The REST interface provides the following API operations for transactional mailings:

- **Get API on page 10**
- **List API on page 11**
- **Txnsend API on page 13**

Get API

URI	`rest/v1/mailings/transactional/{mailing_id}`
Description	The following are the parameters available for this command: • `mailing_id`—Specify the mailing ID.
Request Type	GET

Sample Request	`curl [Authentication ] -k`
	`'https://10.44.51.15/sm/rest/v1/mailings/transactional/101'`

Sample Response	`{`
	`  "asset_id": 112,`
	`  "asset_name": "TxMailing_001",`
	`  "modified_time": "2015-09-14T20:17:16",`
	`  "version_number": 2,`
	`  "approved": false,`
	`  "asset_url": "/sm/rest/v1/mailings/transactional/112",`
	`  "campaign": {`
	`     "asset_id": 100,`
	`     "asset_name": "MyCampain",`
	`     "modified_time": "2015-09-04T20:40:00",`
	`     "version_number": 0`
	`     },`
	`  "channel": "Email",`
	`  "compliance": false,`
	`  "mailing_class": "Default",`
	`  "mailing_priority": "High",`
	`  "mailing_status": "editing",`
	`  "owner": {`
	`      "asset_id": 1,`
	`      "modified_time": "2015-09-07T13:36:44",`
	`      "first_name": "admin",`
	`      "last_name": "Emp"`

	`},` `"planned_launch_date": "2015-09-14",` `"schema": [ "id","email_address", "name" ],` `"target": { "asset_id": 100, "asset_name": "MyTarget_001" }` `}`
Failure Response	`{"errors":["Mailing (id: 1011) is missing/deleted."],"field":null,"more_info":"Check if resource exists","code":400}`

List API

URI	`rest/v1/mailings/transactional/?page=1&limit=100&order=name&sort=asc&search_field=f&search_type=t&search_value=v"`
Description	The following are the parameters available for this command: • `page`—(Required) Specify the starting page number. • `limit`—Specify the page limit. • `order`—Specify page sort criteria based on column names in the Data Source. • `sort`—Specify the page sort order as either `asc` (ascending) or `dsc` (descending). • `search_field`—Enter `asset_name` to search by mailing name. • `search_type`—Specify `is_exactly`, `begins_with`, `ends_with`, or `contains`. • `search_value`—Specify a text string to match in the search. • `status`—Specify a text string to filter mailings based on the specified status.
Request Type	GET
Sample Request	`curl [Authentication] -k 'https://10.44.51.15/sm/rest/v1/mailings/transactional/?page=1&limit=100&order=name&sort=asc'`
Sample Response	`[` `{` `"asset_id": 112,` `"asset_name": "Union",` `"modified_time": "2015-09-14T20:17:16",` `"version_number": 2,` `"approved": false,` `"asset_url": "/sm/rest/v1/mailings/transactional/112",` `"campaign": {` `"asset_id": 100,` `"asset_name": "MyCampain",` `"modified_time": "2015-09-04T20:40:00",`

```
    "version_number": 0
    },
  "channel": "Email",
  "compliance": false,
  "mailing_class": "Default",
  "mailing_priority": "High",
  "mailing_status": "editing",
  "owner": {
    "asset_id": 1,
    "modified_time": "2015-09-07T13:36:44",
    "first_name": "admin",
    "last_name": "Emp"
    },
  "planned_launch_date": "2015-09-14",
  "target": {
    "asset_id": 100,
    "asset_name": "target_cxf_20150904161810"
    }
  },
{
  "asset_id": 133,
  "asset_name": "Xray",
  "modified_time": "2015-09-10T19:37:39",
  "version_number": 6,
  "approved": true,
  "asset_url": "/sm/rest/v1/mailings/transactional/133",
  "channel": "Email",
  "compliance": true,
  "mailing_class": "Default",
  "mailing_priority": "High",
  "mailing_status": "active",
  "owner": {
    "asset_id": 1,
    "modified_time": "2015-09-07T13:36:44",
    "first_name": "admin",
    "last_name": "Emp"
  },
  "target": {
  "asset_id": 103,
  "asset_name": "EDS001"
  }
 }
]
```

Failure Response	`1: {`

	"errors":["Failed to convert value of type 'java.lang.String' to required type 'java.lang.Long'; nested exception is java.lang.NumberFormatException: For input string: \"first\""],
	"field":null,
	"more_info":"Something bad happened classorg.springframework.beans.TypeMismatchException","code":700
	}
	2:{
	"errors":["Required Long parameter 'start' is not present"],
	"field":null,
	"more_info":"You forgot something!",
	"code":700
	}

Txnsend API

URI	`rest/v1/mailings/transactional/{mailing_id}/records -X POST -d '<data>'`
Description	The following are the parameters available for this command: • `mailing_id`—Specify the mailing ID. • `<data>`—Specify data to be sent enclosed in single quotes.
Request Type	POST

Sample Request	{
	"records": [
	{
	"record": {
	"id": "1",
	"email_address": "a@b.com",
	"name": "test_user1"
	}
	},
	{
	"record": {
	"id": "2",
	"email_address": "b@b.com",
	"name": "test_user2"
	}
	},
	{
	"record": {
	"id": "3",
	"email_address": "c@b.com",
	"name": "test_user3"
	}

	`    }`
	`  }`
	`}`

Sample Response	`'Sent successfully'`

Failure Responses	`{"errors":["Mailing (id: 1011) is missing/deleted."],"field":null,"more_info":"Check if resource exists","code":400}`
	`{"errors":["Mailing is not in active status."],"field":null,"more_info":"","code":700}`

Data Source API Operations

The REST interface provides the following APIs for standard batch and transactional mailings:

- **Get API on page 15**
- **List API on page 16**
- **Upsert API (Single Record) on page 17**
- **Import/Upsert API (Bulk Records) on page 18**

Get API

URI	`rest/v1/data-sources/{datasource_type/{id}`
Description	The following are the parameters for this command: • `datasource_type`—Specify data source as one of the following: `internal` or `extension`. • `id`—Specify the data source ID.
Request Type	GET
Sample Request	`curl [Authentication ] -k 'https://10.44.51.15/sm/rest/v1/data-sources/internal/101'`

Sample Response	`{`
	`  "asset_id":101,`
	`  "asset_name":"IDS_CIS",`
	`  "modified_time":"2015-07-29T16:20:47",`
	`  "version_number":25,`
	`  "asset_url":"/sm/rest/v1/data-sources/internal/101",`
	`  "created_on":"2015-06-15T08:50:20",`
	`  "data_source_stat":`
	`     {"num_unsub_rec":0,`
	`      "num_malformed_rec":0,`
	`      "num_duplicate_rec":0,`
	`      "num_total_rec":7,`
	`      "num_invalid_rec":0,`
	`      "num_delta_rec":1,`
	`      "num_update_rec":0,`
	`     "last_import_status":"SUCCESS"},`
	`  "data_source_type":"INTERNAL",`
	`  "department":`
	`    {"asset_id":1,`
	`     "asset_name":"admin",`

	`"asset_url":null},`
	`"description":"Updating description using REST API",`
	`"owner_employee":`
	`{"asset_id":1,`
	`"modified_time":"2015-03-15T09:19:53",`
	`"asset_url":null},`
	`"sync_to_omni":true`
	`}`

Failure Response	`{`
	`"errors":["DataSource (id: 1011) is missing/deleted."],`
	`"field":null,`
	`"more_info":"Check if resource exists",`
	`"code":400`
	`}`

List API

URI	`rest/v1/data-sources/{datasource_ type}/?page=1&limit=100&order=name&sort=asc&search_field=f& search_type=t&search_value=v"`
Description	The folllowing are the parameters for this command: • `datasource_type`—Specify data source as one of the following: `internal` or `extension`, . • `page`—(Required) Specifies the starting page number . • `limit`—Specifies the page limit. • `order`—Specifies page sort criteria. • `sort`—Specifies the page sort order, either `asc` (ascending) or `dsc` (descending). • `search_field`—Enter `asset_name` to search by data source name. • `search_type`——Specify `is_exactly`, `begins_with`, `ends_with`, or `contains`. • `search_value`—Specify a text string to match in the search.
Request Type	GET
Sample Request	`curl [Authentication] -k 'https://10.44.51.15/sm/rest/v1/data-sources/internal?page=1&limit=5&order=name&sort=dsc'`

Sample Response	`[`
	`{`
	`"asset_id":103,`
	`"asset_name":"IDS_001",`
	`"modified_time":"2015-06-12T14:46:23",`
	`"version_number":2,`

	"asset_url":"/sm/rest/v1/data-sources/internal/103",
	"data_source_stat":{"num_total_rec":1}
	},
	{
	"asset_id":108,
	"asset_name":"IDS_010",
	"modified_time":"2015-06-12T20:00:23",
	"version_number":1,
	"asset_url":"/sm/rest/v1/data-sources/108",
	"data_source_stat":{"num_total_rec":1}
	}
	]

Failure Response	1: {
	"errors":["Failed to convert value of type 'java.lang.String' to required type 'java.lang.Long'; nested exception is java.lang.NumberFormatException: For input string: \"first\""],
	"field":null,
	"more_info":"Something bad happened class org.springframework.beans.TypeMismatchException","code":700
	}
	2:{
	"errors":["Required Long parameter 'start' is not present"],
	"field":null,
	"more_info":"You forgot something!",
	"code":700
	}
	3: {
	"errors":["Only one record can be upserted at a time"],
	"field":null,
	"more_info":"Something bad happened!",
	"code":700
	}

Upsert API (Single Record)

URI	`rest/v1/data-sources/{datasource-type}/{id}/records -X PUT`
Description	The following are the parameters for this command: • `datasource-type`—Specify either `internal` or `extension`. • `id`—Specify the data source ID.
Request Type	PUT

Java Bean Representation	`List<List<String>> data;`
	`List<String> header;`

Sample Request	{ "data": [

	`[9999", 1]`
	`],`
	`"header": ["order_id","customer_id"]`
	`}`

Sample Response	`{`
	`"insert count": 1,`
	`"update_count": 0,`
	`"malformed_count": 0`
	`}`

Failure Response	`1: {`
	`"errors":[ "Datasource (id: 5466) is missing/deleted."],`
	`"field":null,"more_info":"Check if resource exists",`
	`"code":400`
	`}`
	`2:{`
	`"errors":["Invalid column name specified in import file: id1"],`
	`"field":null,"more_info":"Something bad happened!",`
	`"code":600`
	`}`
	`3: {`
	`"errors":["Only one record can be upserted at a time"],`
	`"field":null,`
	`"more_info":"Something bad happened class java.lang.IllegalStateException",`
	`"code":700`
	`} '`

Import/Upsert API (Bulk Records)

URI	`rest/v1/data-sources/{datasource-type}/{id}/upload`
Description	The following are the parameters for this command: • `datasource-type`—Specify either `internal` or `extension`. • `id`—Specify the data source ID.
Request Type	POST
Sample Request	**File Upload Option:** `curl -H 'Authorization:xxx' -H 'X-Organization: admin' -H "Content-type: multipart/form-data;" -X POST -F "meta-data=@import-meta.txt;type=application/json" -F "file-data=@text-data.txt" https://127.0.0.1/sm/rest/v1/data-sources/internal/8555/upload -k -1`

Content of file "/tmp/import-meta-data.txt"

```
{
   "mode" : "UPSERT_RECORDS",
   "column_map": {
       "id": 1,
       "email": 2 },
   "first_row_header":true,
   "delimiter": ","
}
```

HTTPS Upload with meta data in the request:

```
curl -H 'Authorization:xxx' -H 'X-Organization: admin' -H
"Content-type: multipart/form-data;" -X POST -F 'meta-data=
{"mode" :"ADD_RECORDS","first_row_header":false,"delimiter":
","};type=application/json' -F "file-data=@text-data.txt"
https://127.0.0.1/sm/rest/v1/data-sources/internal/855/upload -k
-1
```

Note: The file **text-data.txt** is expected to contain the records for the bulk operation.

Content of file "text-data.csv"

```
"id", "email"
"9999", "testRESTAPI9999@local-test.strongmailsystems.com"
"9998", "testRESTAPI9998@local-test.strongmailsystems.com"
```

FTP Upload Option:

```
curl -H 'Authorization:xxx' -H 'X-Organization: admin' -H
"Content-type: multipart/form-data;" -X POST -F " 'meta-data=
{"mode" :"ADD_RECORDS","first_row_header":true,"delimiter":
","};type=application/json' -F "ftp-file-name=@my_ftp.txt"
https://127.0.0.1/sm/rest/v1/data-sources/internal/855/upload -k
-1
```

Supported Import Modes:

- **ADD_RECORDS**—New records will be added to the data source and existing or duplicate records (based on Primary Key of data source) will be ignored.
- **UPSERT_RECORDS**—New records will be added to the datasource and existing or duplicate records (based on Primary Key of data source) will be updated.
- **REPLACE_RECORDS**—The records currently present in the data source will be discarded and the records specified in the request will be added.

Meta-data parameters:

	• **mode**—Import operation mode. (ADD_RECORDS / UPSERT_ RECORDS) • **Column map**— (Optional) User can specify the position/index of the columnwhere the data resides in the data source, instead of providing the column name. Recommended to use when column headers are not present in the data source. • **First_row_header**—(Optional) Specifies whether to include (true) column headers in the data source. If set to 'false', the column headers are not include. • **delimiter**—(Optional) Delimiter used in the data source between columns. (Default: ',') **FTP Upload with meta data in the request:** `curl [Authentication] -H "Content-type: multipart/form-data;" -X POST -F 'meta-data={"mode" : "ADD_RECORDS","is_first_row_ header":false,"delimiter": ","};type=application/json' -F "ftp-file-name=my_ftp.txt" https://10.44.51.15/sm/rest/v1/data-sources/internal/113/upload -k` **Note:** The file **my_ftp.txt** must be present in the *ftp-upload* directory of the logged-in organization.
Sample Response	`{"message":"Importing file for data source <id>"}`

Failure Response	`{`
	`"errors":[ "Datasource (id: <id>) is missing/deleted."],`
	`"field":null,"more_info":"Check if resource exists",`
	`"code":400`
	`}`

Program API Operations

The REST interface provides the following operations for Lifecycle Marketing Programs:

- **Get API on page 21**
- **List API on page 22**
- **AddtoProgram API on page 25**

Get API

URI	`rest/v1/programs/{program_id}`
Description	The following are the parameters available for this command: • `program_id`—Specify the Lifecycle Marketing program ID.
Request Type	GET
Sample Request	`curl [Authentication ] -k 'https://10.44.51.15/sm/rest/v1/mailings/transactional/101'`

Sample Response	`{`
	`  "asset_name": "P1",`
	`  "modified_time": "2015-09-05T21:09:43",`
	`  "version_number": 1,`
	`  "asset_url": "/sm/rest/v1/programs/100",`
	`  "primary_source": {`
	`      "asset_id": 101,`
	`      "asset_name": "P1",`
	`      "modified_time": "2015-09-05T21:09:06",`
	`      "version_number": 0,`
	`      "asset_url": "/sm/rest/v1/data-sources/101",`
	`      "data_source_type": "PROGRAMS"`
	`     },`
	`  "primary_source_schema": [`
	`     {`
	`      "field_name": "id",`
	`      "field_data_type": "Integer"`
	`     },`
	`     {`
	`      "field_name": "name",`
	`      "field_data_type": "Text"`
	`     },`

```
     {
     "field_name": "email",
     "field_data_type": "Email Address"
    },
    {
     "field_name": "unsub",
     "field_data_type": "Text"
    }
   ],
  "program_data_source": {
     "asset_id": 101,
     "asset_name": "P1",
     "modified_time": "2015-09-05T21:09:06",
     "version_number": 0,
     "asset_url": "/sm/rest/v1/data-sources/101",
     "data_source_type": "PROGRAMS"
    },
  "program_status": "Editing",
  "start_type": "Targeted Start"
}
```

Failure Response	`{"errors":["Program (id: 800) is missing/deleted."],"field":null,"more_info":"Check if resource exists","code":400}`

List API

URI	`rest/v1/programs/?page=1&limit=100&order=name&sort=asc&search_field=name&search_type=contains&search_value=PG"`
Description	The following are the parameters available for this command: • `page`—(Required) Specify the starting page number. • `limit`—Specify the page limit. • `order`—Specify page sort criteria, based on column names in the Program Data Source. • `sort`—Specify the page sort order, either `asc` (ascending) or `dsc` (descending). • `search_field`—Enter `asset_name` to search by program name. • `search_type`—Specify `is_exactly`, `begins_with`, `ends_with`, or `contains`. • `search_value`—Specify a text string to match in the search. • `status`—Specify a text string to filter mailings based on the specified status.

	• `type`—Specify a text string to filter mailings based on the specified type of the Start node. • `approved`—Specify a Boolean string to filter programs based on whether or not they are approved.
Request Type	GET
Sample Request	`url [Authentication] -k 'https://10.44.51.15/sm/rest/v1/programs?page=1&limit=100&order=name&sort=asc'`

Sample Response

```
  {
       "asset_id": 101,
       "asset_name": "P2",
       "modified_time": "2015-09-05T21:20:32",
       "version_number": 1,
       "asset_url": "/sm/rest/v1/programs/101",
       "primary_source": {
           "asset_id": 101,
           "asset_name": "P1",
           "modified_time": "2015-09-05T21:09:06",
           "version_number": 0,
           "asset_url": "/sm/rest/v1/data-sources/101",
           "data_source_type": "PROGRAMS"
       },
       "program_data_source": {
           "asset_id": 101,
           "asset_name": "P1",
           "modified_time": "2015-09-05T21:09:06",
           "version_number": 0,
           "asset_url": "/sm/rest/v1/data-sources/101",
           "data_source_type": "PROGRAMS"
       },
       "program_status": "Editing",
       "start_type": "Targeted Start"
   },
   {
       "asset_id": 100,
       "asset_name": "P1",
       "modified_time": "2015-09-05T21:09:43",
       "version_number": 1,
       "asset_url": "/sm/rest/v1/programs/100",
       "primary_source": {
           "asset_id": 101,
           "asset_name": "P1",
           "modified_time": "2015-09-05T21:09:06",
           "version_number": 0,
           "asset_url": "/sm/rest/v1/data-sources/101",
           "data_source_type": "PROGRAMS"
       },
       "program_data_source": {
           "asset_id": 101,
           "asset_name": "P1",
           "modified_time": "2015-09-05T21:09:06",
           "version_number": 0,
           "asset_url": "/sm/rest/v1/data-sources/101",
           "data_source_type": "PROGRAMS"
       },
       "program_status": "Editing",
       "start_type": "Targeted Start"
   }
```

Failure Responses

```
{
  "errors":["Failed to convert value of type 'java.lang.String'
   to required type 'java.lang.Long'; nested exception is
   java.lang.NumberFormatException: For input string: \"first\""],
  "field":null,
  "more_info":"Something bad happened.",
```

	`"code":700`
	`}`
	`{`
	`"errors":["Required Long parameter 'start' is not present"],`
	`"field":null,`
	`"more_info":"You forgot something!",`
	`"code":700`
	`}`

AddtoProgram API

URI	`rest/v1/programs/{program_id}/records -X POST -d '<data>'`
Description	This API performs the following two functions in a single service request: The following are the parameters available for this command: • `program_id`—Specifies the program ID. • `<data>`—Specify data to be sent enclosed in single quotes.
Request Type	POST
Java Bean Representation	`List<Map<String,String>> records;`

Sample Request	`curl [Authentication ]-k https://10.44.51.15/sm/rest/v1/programs/101/records' -X POST`
	`{`
	`"records": [`
	`{`
	`"record": {`
	`"id": "1",`
	`"email_address": "a@b.com",`
	`"name": "test_user1"`
	`}`
	`},`
	`{`
	`"record": {`
	`"id": "2",`
	`"email_address": "b@b.com",`
	`"name": "test_user2"`
	`}`
	`},`
	`{`
	`"record": {`
	`"id": "3",`
	`"email_address": "c@b.com",`
	`"name": "test_user3"`
	`}`

	```
	    }
	  ]
	}
	```

Sample Request	
	```
	{
	  "success": true,
	  "success_count": 10,
	  "failure_count": 3,
	  "duplicate_count": 2,
	}
	```