

selligent

MESSAGE STUDIO 9.1

REST API Reference Guide

Copyright © 2016 Selligent, Inc. All rights reserved.

No part of the contents of this publication may be reproduced or transmitted in any form or by any means without the written permission of Selligent, Inc.

The information furnished herein is believed to be accurate and reliable. However, no responsibility is assumed by Selligent, Inc. for its use, or for any infringements of third party rights resulting from its use.

Selligent and the Selligent logo are registered trademarks in the United States, other countries, or both. All rights reserved.

Wednesday, March 23, 2016

Contents

Contents	3
Introduction	4
REST API Framework	5
URI Structure	5
HTTP Methods	5
API Authentication	6
CRUD Operations	7
Versioning & Response Format	7
Mailing API Operations	8
Data Source API Operations	13
Program API Operations	17

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](#)**
- **[Mailing API Operations on page 8](#)**
- **[Data Source API Operations on page 13](#)**
- **[Program API Operations on page 17](#)**

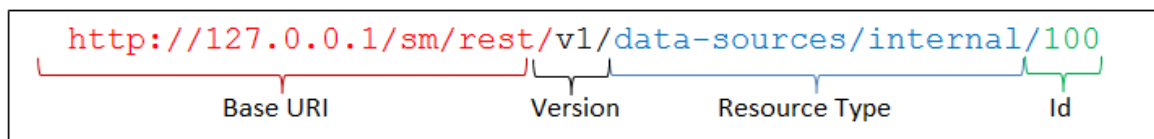
REST API Framework

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.

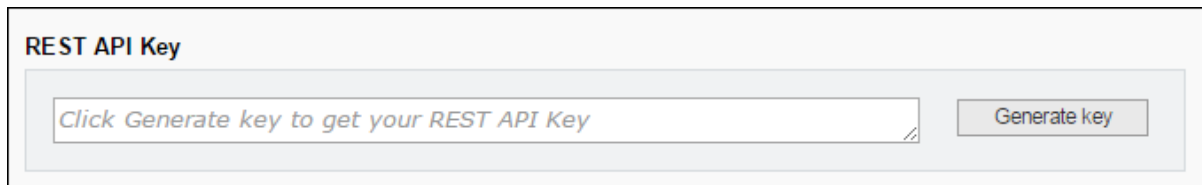
API Authentication

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



The screenshot shows a user interface for generating a REST API key. At the top, the text "REST API Key" is displayed. Below it, there is a large text input field containing the placeholder text "Click Generate key to get your REST API Key". To the right of this field is a button labeled "Generate 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>
```

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.

Mailing API Operations

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

- [Get API on page 8](#)
- [List API on page 9](#)
- [Txnsend API on page 11](#)

Get API

URI	rest/v1/mailings/{mailing_type}/{mailing_id}
Description	<p>The following are the parameters available for this command:</p> <ul style="list-style-type: none">• mailing_type—The only valid value is transactional.• mailing_id—Specify the mailing ID.
Request Type	GET
Sample Request	<pre>curl [Authentication] -k 'https://10.44.51.15/sm/rest/v1/mailings/transactional/101'</pre>
Sample Response	<pre>{ "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,</pre>

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

List API

URI	<pre> rest/v1/mailings/{mailing_type} /?page=1&limit=100&order=name&sort=asc&search_field=f& search_type=t&search_value=v" </pre>
Description	<p>The following are the parameters available for this command:</p> <ul style="list-style-type: none"> • page—(Required) Specify the starting page number. • mailing_type—May be either <code>transactional</code> or <code>batch</code>. • 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 <code>asc</code> (ascending) or <code>dsc</code> (descending). • search_field—Enter <code>asset_name</code> to search by mailing name. • search_type—Specify <code>is_exactly</code>, <code>begins_with</code>, <code>ends_with</code>, or <code>contains</code>. • 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	<pre> curl [Authentication] -k 'https://10.44.51.15/sm/rest/v1/mailings/transactional/?page=1 &limit=100&order=name&sort=asc' </pre>
Sample Response	<pre> [{ "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", </pre>

```

"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	<pre> 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 } </pre>

Txnsend API

URI	rest/v1/mailings/transactional/{mailing_id}/records -X POST -d '<data>'
Description	<p>The following are the parameters available for this command:</p> <ul style="list-style-type: none"> mailing_id—Specify the mailing ID. <data>—Specify data to be sent enclosed in single quotes.
Request Type	POST
Sample Request	<pre> { "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", </pre>

	<pre> "name": "test_user3" } } }</pre>
Sample Response	'Sent successfully'
Failure Responses	<pre>{"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}</pre>

Data Source API Operations

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

- [Get API on page 13](#)
- [List API on page 14](#)
- [Data Source API Operations on page 13](#)
- [Bulk Import API on page 15](#)

Get API

URI	rest/v1/datasources/{datasource_type}/{id}
Description	<p>The following are the parameters for this command:</p> <ul style="list-style-type: none">• <code>datasource_type</code>—Specify data source as one of the following: <code>internal</code> or <code>extension</code>.• <code>id</code>—Specify the data source ID.
Request Type	GET
Sample Request	<pre>curl [Authentication] -k 'https://10.44.51.15/sm/rest/v1/datasources/internal/101'</pre>
Sample Response	<pre>{ "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",</pre>

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

List API

URI	rest/v1/datasources/{datasource_type} /?page=1&limit=100&order=name&sort=asc&search_field=f&search_type=t&search_value=v"
Description	<p>The following are the parameters for this command:</p> <ul style="list-style-type: none"> • 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	<pre> [{ "asset_id":103, "asset_name":"IDS_001", "modified_time":"2015-06-12T14:46:23", "version_number":2, </pre>

	<pre> "asset_url":"/sm/rest/v1/datasources/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} }] </pre>
Failure Response	<pre> 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 } </pre>

Bulk Import API

URI	rest/v1/data-sources/{datasource-type}/{id}/upload -X POST
Description	<p>The following are the parameters for this command:</p> <ul style="list-style-type: none"> datasource-type—Specify either internal or extension. id—Specify the data source ID.
Request Type	POST
Sample Request	<p>HTTPS Upload:</p> <pre> curl [Authentication] -H "Content-type: multipart/form-data;" -X POST -F "meta-data=@import-meta.txt;type=application/json" -F "file-data=@ds/data.txt" https </pre>

	<p>HTTPS Upload with meta data in the request:</p> <pre>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 "file-data=@ds/data.txt" https://10.44.51.15/sm/rest/v1/data-sources/internal/113/upload -k mode=ADD_RECORDS,UPSERT_RECORD</pre> <hr/> <p>Note: The file data.txt is expected to contain the records for the bulk operation.</p> <hr/> <p>FTP Upload:</p> <pre>curl [Authentication] -H "Content-type: multipart/form-data;" -X POST -F "meta-data=@import-meta.txt;type=application/json" -F "ftp-file-name=my_ftp.txt" https://10.44.51.15/sm/rest/v1/data-sources/internal/113/upload -k</pre> <hr/> <p>FTP Upload with meta data in the request:</p> <pre>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</pre> <hr/> <p>Note: The file my_ftp.txt is expected to be present in the FTP upload folder.</p>
Sample Response	<pre>{"message":"Importing file for data source <id>"}</pre>
Failure Response	<pre>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": ["<file name> is not found in FTP folder"], "field": null, "more_info": "Internal Error", "code": 600 }</pre>

Program API Operations

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

- [Get API on page 17](#)
- [List API on page 18](#)
- [AddtoProgram API on page 21](#)

Get API

URI	rest/v1/programs/{program_id}
Description	The following are the parameters available for this command: <ul style="list-style-type: none">• program_id—Specify the Lifecycle Marketing program ID.
Request Type	GET
Sample Request	<pre>curl [Authentication] -k 'https://10.44.51.15/sm/rest/v1/mailings/transactional/101'</pre>
Sample Response	<pre>{ "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" }] }</pre>

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

List API

URI	rest/v1/programs/?page=1&limit=100&order=name&sort=asc&search_field=name&search_type=contains&search_value=PG"
Description	<p>The following are the parameters available for this command:</p> <ul style="list-style-type: none"> • 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 <i>asc</i> (ascending) or <i>dsc</i> (descending). • search_field—Enter <i>asset_name</i> to search by program name. • search_type—Specify <i>is_exactly</i>, <i>begins_with</i>, <i>ends_with</i>, or <i>contains</i>. • search_value—Specify a text string to match in the search. • status—Specify a text string to filter mailings based on the specified status.

	<ul style="list-style-type: none">• <code>type</code>—Specify a text string to filter mailings based on the specified type of the Start node.• <code>approved</code>—Specify a Boolean string to filter programs based on whether or not they are approved.
Request Type	GET
Sample Request	<code>url [Authentication] -k 'https://10.44.51.15/sm/rest/v1/programs?page=1&limit=100&order=name&sort=asc'</code>

Sample Response	<pre> { "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" } </pre>
Failure Responses	<pre> { "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.", } </pre>

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

AddtoProgram API

URI	rest/v1/programs/{program_id}/records -X POST -d '<data>'
Description	<p>This API performs the following two functions in a single service request:</p> <ul style="list-style-type: none"> • Inserts the record into the Primary data source of the program. The primary source of the program could be a Program Data Source (PDS) or Extension. The record will be updated if it is already present in the primary data source of the program. • Adds the record into the program (LCM). <p>The following are the parameters available for this command:</p> <ul style="list-style-type: none"> • 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	<pre> 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" } }], } </pre>

	<pre>{ "record": { "id": "3", "email_address": "c@b.com", "name": "test_user3" } }</pre>
Sample Request	<pre>{ "success": true, "success_count": 10, "failure_count": 3, "duplicate_count": 2, }</pre>