Oracle Responsys REST API

Developer's Guide

Release 6.26

Contents

Overview 1

Processing Responsys REST API requests 1

Authenticating 2

- Login with username and password 2
- Login with username and certificates 2
- Refresh token 3

Managing Profile List Tables 4

- Merge or update members in a profile list table 4
- Retrieve a member of a profile list using RIID 6
- Retrieve a member of a profile list based on query attribute 7

Managing Profile Extension Tables 9

- Retrieve all profile extensions of a profile list 9
- Create a new profile extension table 11
- Merge or update members in a profile extension table 11
- Retrieve a member of a profile extension table based on RIID 13
- Retrieve a member of a profile extension table based on a query attribute 15
- Delete a member of a profile extension table based on RIID 16

Managing Supplemental Tables 18

- Create a new supplemental table 18
- Merge supplemental table records using primary key 18
- Merge supplemental table records without primary key 21
- Retrieve supplemental table records with primary key 25
- Delete supplemental table records 26

Triggering Email Messages 28

- Merge members into a profile list and trigger email messages to them 28
- Trigger email message 30

Triggering SMS Messages 31

- Merge members into a profile list and trigger SMS messages to them 31

Raising Custom Events for Cross-channel Marketing Programs 32

- Trigger a custom event 32

Managing Email Campaigns 33
Create email campaign 33
Update email campaign 36
Get email campaign 39
Schedule email campaign 41
Get email campaign schedule 42
Update email campaign schedule 43
Delete email campaign schedule 44
Get all email campaigns 45
Managing Content Library Folders 49
Create content library folder 49
Delete content library folder 50
List contents of a content library folder 50
Managing Content Library Documents 53
Create content library document 53
Retrieve contents of a content library document 54
Update contents of a content library document 55
Delete a content library document 56
Create a copy of a content library document 57
Managing Content Library Media Files 58
Create content library media file 58
Retrieve contents of a content library media file 59
Update contents of a content library media file 61
Delete a content library media file 63
Create a copy of a content library media file 63
Managing Images of Content Library Documents 65
Set images in a content library document 65
Get images in a content library document 66
Handling Errors 68
Definitions of Rule Parameters for Merging Members into a Profile List 74
Campaign Object Attributes 75

Overview

This document provides a guide for software developers to use the Responsys REST API.

This release supports several resources including: profile lists, profile extensions, supplemental data, campaigns, events, content library folders, content library documents, content library media files, and content library document images.

Responsys REST APIs are JSON-aware for accepting or returning a payload. These REST APIs also comply with the HATEAOS principle such that a client interacts with a network application entirely through hypermedia provided dynamically by application servers. Therefore, a REST client needs no prior knowledge about how to interact with any particular application or server beyond a generic understanding of hypermedia. As a result, the response payloads returned by most of the Responsys REST interfaces contain additional information (specifically "links") to allow the client application to transition through application states. More information about HATEAOS is available at https://en.wikipedia.org/wiki/HATEOAS.

Responsys SOAP and REST API are two separate sets of APIs, but they use the same underlying object model. To learn more about REST vs. SOAP, please visit http://www.slideshare.net/Muratakal/rest-vs-soap-15854355

Processing Responsys REST API requests

The sequence of steps required for processing the Responsys REST API requests is:

1. Client issues an HTTP POST request to authenticate via the login endpoint. Depending on the pod that hosts the Responsys account the end points could be either `login2.responsys.net` (for interact2 pod) or `login5.responsys.net` (for interact5 pod).
2. A JSON response is returned with a token and an API endpoint to be used for subsequent API requests.
3. Client issues desired HTTP POST to the API endpoint along with the token in the HTTP HEADER.
   Note that the API endpoint must be crafted from the URL endpoint of step 2 and the specific path of the desired API.
4. If the API request is successfully processed, a specific JSON response is returned according to the specification of the processed API. Otherwise, an error payload is returned for interpretation.
5. Depending on the success or failure of the previous API request, take the next action.
6. Repeat step 3-5 as needed.
7. If needed, refresh the token to avoid having to re-authenticate.
   By default, tokens last for two hours.

Authenticating

The very first REST API request must be to authenticate to a specific Responsys account using a username and a password or certificates. Upon successful authentication, a token and an endpoint are returned that you must use for any subsequent REST API request.

Login with username and password

Important Note: For security reasons you must pass username and password as parameters in the POST request body only with the correct content type ('Content-Type': 'application/x-www-form-urlencoded'). Therefore, you must NOT use the URL string for passing in the authentication credentials.

Service URL:

```
/rest/api/v1/auth/token
```

Request Method:

```
POST
```

Request Parameters:

```
user_name=<USER_NAME>
password=<PASSWORD>
auth_type=password
```

Sample Login Request:

URL:

```
/rest/api/v1/auth/token
```

Sample Request Body:

```
user_name=<USER_NAME>&password=<PASSWORD>&auth_type=password
```

Response:

```
{
 "authToken" : "<AUTH_TOKEN>",
 "issuedAt" :  < TIMESTAMP > ,
 "endPoint" : "<ENDPOINT_URI>"
}
```

Login with username and certificates

Instead of using a password, a server and a client certificate can be used for authenticating with a username. The following two REST requests must be processed in sequence for authenticating with certificates:

1. Service URL:

```
/rest/api/v1/auth/token
```

Request Method:

```
POST
```

Request Parameters:

```
user_name=<USER_NAME>
auth_type=server
client_challenge=<BASE_64_ENCODED_CLIENT_CHALLENGE>
```

Response:

```
{
 "authToken" : "<TEMP_AUTH_TOKEN>",
 "serverChallenge" : "<BASE_64_ENCODED_SERVER_CHALLENGE>",
 "clientChallenge" :
"<ENCRYPTED_AND_THEN_BASE_64_ENCODED_CLIENT_CHALLENGE>"
}
```

2. Service URL:

```
/rest/api/v1/auth/token
```

Request Method:

```
POST
```

Request Parameters:

```
user_name=<USER_NAME>
auth_type=client
server_challenge=<ENCRYPTED_AND_THEN_BASE_64_ENCODED_SERVER_CHALLENGE>
```

Request Header:

```
 Authorization=<TEMP_AUTH_TOKEN>
```
(this is obtained from the above call)

Response:

```
{
 "authToken" : "<AUTH_TOKEN>",
 "issuedAt" :  <TIMESTAMP > ,
 "endPoint" : "<ENDPOINT_URI>"
}
```

Refresh token

A token expires after a certain period of time (by default, two hours). An existing token can be refreshed within that period without having to authenticate again.

Service URL:

```
/rest/api/v1/auth/token
```

Request Method:

```
POST
```

Request Parameters:

```
auth_type=token
```

Request Header:

```
 Authorization=<AUTH_TOKEN>
```

Response:

```
{
 "authToken" : "<AUTH_TOKEN>",
 "issuedAt" :  <TIMESTAMP > ,
 "endPoint" : "<ENDPOINT_URI>"
}
```

Managing Profile List Tables

New members can be added to a list or attribute values of existing members can be updated. Also, members of a list can be retrieved.

Merge or update members in a profile list table

New members can be added to an existing profile list and existing members in a profile list can be updated. For a given list in a specific folder, an array of record data that contain field names and their corresponding field values are specified. Please see the definition of merge rule parameters provided at the end of this document. Up to 200 members can be handled per a single request.

Service URL:

```
/rest/api/v1/lists/<listName>/members
```

Request Method:

```
POST
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request JSON Body:

```
{
        "recordData" : {
                "fieldNames" : ["riid_", "mobile_number_", "email_address_"],
                "records" :   [
                                ["4094326", "9845349498","ab.cd@gmail.com"],
                                ["4094327", "9844444444","unknown@oracle.com"],
                                ["4094328", "9844444666","abc@gmail.com"],
                                ["ssdcf", "9844444444","xyz"]
                ],
                "mapTemplateName" : null
        },
        "mergeRule" : {
                "htmlValue" : "H",
                "optinValue" : "I",
                "textValue" : "T",
                "insertOnNoMatch" : true,
                "updateOnMatch" : "REPLACE_ALL",
                "matchColumnName1" : "RIID_",
                "matchColumnName2" : null,
                "matchOperator" : "NONE",
                "optoutValue" : "O",
                "rejectRecordIfChannelEmpty" : null,
                "defaultPermissionStatus" : "OPTIN"
        }
}
```

Sample response in case of success

NOTE: Irrespective of what field names were used to perform the merge, the response will always contain only 'RIID_" in the 'fieldNames" attribute and the corresponding "RIID_" values for the records in the "records" attribute. In case merge failed for a record, the "RIID_" of the record is not present in the

response. Instead, an error message starting with "MERGEFAILED:" is returned. Client developers can look for the string "MERGEFAILED:" in the response for a particular row to determine whether that recipient was merged successfully or not. Also note that the order of records in the response matches the order of records specified in the request payload. Furthermore, other attributes in the response like mapTemplateName and mergeRule will mirror the valid values specified in the request payload or default to null/false in case of invalid values.


```
{
   "recordData":    {
      "fieldNames": ["RIID_"],
      "records":    [
         ["4094326"],
         ["4094327"],
         ["4094328"],
         ["MERGEFAILED: Record 3 = INVALID_PARAMETER: The value ssdcf is not
valid for an integer field\n\n\r\n"]
      ],
      "mapTemplateName": null
   },
   "mergeRule":    {
      "textValue": "T",
      "insertOnNoMatch": true,
      "updateOnMatch": "REPLACE_ALL",
      "matchOperator": "NONE",
      "matchColumnName3": null,
      "matchColumnName1": "RIID_",
      "matchColumnName2": null,
      "optinValue": "I",
      "optoutValue": "O",
      "rejectRecordIfChannelEmpty": null,
      "htmlValue": "H",
      "defaultPermissionStatus": "OPTIN"
   },
   "links":    [
      {
         "rel": "self",
         "href": "/rest/api/v1/lists/DemoNewsLetterList/members",
         "method": "POST"
      },
      {
         "rel": "retrieveListRecipientsRIID",
         "href": "/rest/api/v1/lists/DemoNewsLetterList/members/<riid>",
         "method": "GET"
      }
   ]
}
```


Sample response in case of failure


```
{
   "type": "",
   "title": "Invalid request parameters",
   "errorCode": "INVALID_PARAMETER",
   "detail": "matchColumnName1 in ListMergeRule is null or empty",
   "errorDetails": []
}
```


The matchColumn attributes can have the following possible values:

RIID_,
*CUSTOMER_ID_*
*EMAIL_ADDRESS_*
*MOBILE_NUMBER_*
*EMAIL_MD5_HASH_*
*EMAIL_SHA256_HASH_*

Also note that a combination of either of the HASH keys or a combination of EMAIL_ADDRESS_ with either of the HASH_ keys cannot be given as matchColumns at the same time.
Also when either of the HASH_ keys exists as a match column, then only updates are possible, thus the corresponding value in insertOnNoMatch should be set to false when using these columns as matchColumns.

Retrieve a member of a profile list using RIID

Existing members of a profile list can be retrieved one at a time by using the Responsys ID (RIID). Please note that the total length of the string passed in for the fs parameter (containing the comma separated field names) cannot exceed 150 characters. To retrieve values of all columns, you can specify only one field with value set to 'all' (if you have a column called 'all', you should use two or more specific column names to avoid getting all of the columns).

Service URL:

```
/rest/api/v1/lists/<listName>/members/<riid>
```

Request Method:

```
GET
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request parameters:

```
fs - comma separated list of fields to retrieve or 'all'
```

Sample Response in case of success:

NOTE: Other attributes in the response like mapTemplateName and mergeRule will have default values of null/false.


```
{
   "recordData":    {
      "fieldNames":       [
         "RIID_",
         "EMAIL_ADDRESS_",
         "MOBILE_NUMBER_"
      ],
      "records": [      [
         "4094326",
         "ab.cd@gmail.com",
         "9845349498"
      ]],
      "mapTemplateName": null
   },
   "mergeRule":    {
      "textValue": null,
```

```
        "insertOnNoMatch": false,
        "updateOnMatch": null,
        "matchOperator": null,
        "matchColumnName3": null,
        "matchColumnName1": null,
        "matchColumnName2": null,
        "optinValue": null,
        "optoutValue": null,
        "rejectRecordIfChannelEmpty": null,
        "htmlValue": null,
        "defaultPermissionStatus": null
    },
    "links":    [
         {
          "rel": "self",
          "href":
"/rest/api/v1/lists/DemoNewsLetterList/members/4094326?fs=riid_,email_address_,mo
bile_number_",
          "method": "GET"
       },
        {
          "rel": "mergeListRecipients",
          "href": "/rest/api/v1/lists/DemoNewsLetterList/members",
          "method": "POST"
       }
    ]
}
```


Sample Response in case of failure:


```
{
    "type": "",
    "title": "Record not found",
    "errorCode": "RECORD_NOT_FOUND",
    "detail": "No records found in the list for given ids",
    "errorDetails": []
}
```


Retrieve a member of a profile list based on query attribute

Existing members of a profile list can be retrieved one at a time by using a query attribute if the Responsys ID (RIID) for the member is not available. Please note that the total length of the string passed in for the fs parameter (containing the comma separated field names) cannot exceed 150 characters. To retrieve values of all columns, you can specify only one field with value set to 'all' (if you have a column called 'all', you should use two or more specific column names to avoid getting all of the columns).

Service URL:

```
/rest/api/v1/lists/<listName>/members
```

Request Method:

```
GET
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request parameters:

```
qa – Query Attribute. Can be either 'r', 'e', 'c' or 'm'
id – ID corresponding to the query attribute
```

```
fs – Comma separated field list or 'all'
```

Sample Response in case of success:

NOTE: Other attributes in the response like mapTemplateName and mergeRule will have default values of null/false.


```
{
   "recordData":    {
      "fieldNames":       [
         "RIID_",
         "EMAIL_ADDRESS_",
         "CUSTOMER_ID_"
      ],
      "records":       [
         [
            "4094330",
            "ab.na@gmail.com",
            null
         ],
         [
            "4094326",
            "ab.cd@gmail.com",
            null
         ]
      ],
      "mapTemplateName": null
   },
   "mergeRule":    {
      "textValue": null,
      "insertOnNoMatch": false,
      "updateOnMatch": null,
      "matchOperator": null,
      "matchColumnName3": null,
      "matchColumnName1": null,
      "matchColumnName2": null,
      "optinValue": null,
      "optoutValue": null,
      "rejectRecordIfChannelEmpty": null,
      "htmlValue": null,
      "defaultPermissionStatus": null
   },
   "links":    [
      {
         "rel": "self",
         "href":
"/rest/api/v1/lists/DemoNewsLetterList/members?qa=m&fs=riid_,email_address_,custo
mer_id_&id=9845349498",
         "method": "GET"
      },
      {
         "rel": "mergeListRecipients",
         "href": "/rest/api/v1/lists/DemoNewsLetterList/members",
         "method": "POST"
      }
   ]
}
```


Sample Response in case of failure:


```
{
   "type": "",
```

```
    "title": "Invalid field name",
    "errorCode": "INVALID_FIELD_NAME",
    "detail": "Column(s) [CUSTOMER_ID] not found in the list",
    "errorDetails": []
}
```

Managing Profile Extension Tables

For a given profile list table, its profile extension tables can be retrieved. Also new profile extension tables can be created, and for an existing profile extension table, its members can be added, updated, retrieved, or deleted.

Retrieve all profile extensions of a profile list

This interface is to retrieve all profile extension tables (PETs) associated with a given profile list table.

Service URL:

```
/rest/api/v1/lists/<listName>/listExtensions
```

Request Method:

```
GET
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request Parameters:

```
None
```

Sample Response in case of success:

Note: The response is a collection of all PETs for the specified Profile List. Each of the individual objects in the collection represents a Profile Extension Object with a link 'Create a Profile Extension' for the Profile List.

```
[{
   "profileExtension":    {
      "objectName": "WS_Auto_RIPET",
      "folderName": "WS_Auto_RIFolder"
   },
   "fields":    [
      {
         "fieldName": "RIID_",
         "fieldType": "INTEGER"
      },
      {
         "fieldName": "ENAME",
         "fieldType": "STR500"
      },
      {
         "fieldName": "EMPID",
         "fieldType": "STR500"
      },
      {
         "fieldName": "CREATED_BY_LOAD_JOB_ID_",
         "fieldType": "INTEGER"
      },
      {
```

```
            "fieldName": "LAST_MOD_BY_LOAD_JOB_ID_",
            "fieldType": "INTEGER"
         },
         {
            "fieldName": "CREATED_DATE_",
            "fieldType": "TIMESTAMP"
         },
         {
            "fieldName": "MODIFIED_DATE_",
            "fieldType": "TIMESTAMP"
         },
         {
            "fieldName": "LAST_BULK_LOAD_ID_",
            "fieldType": "INTEGER"
         },
         {
            "fieldName": "EMAIL_SHA256_HASH_",
            "fieldType": "STR100"
         },
         {
            "fieldName": "EMAIL_ADDRESS_",
            "fieldType": "STR500"
         },
         {
            "fieldName": "CUSTOMER_ID_",
            "fieldType": "STR255"
         },
         {
            "fieldName": "EMAIL_PERMISSION_STATUS_",
            "fieldType": "CHAR"
         },
         {
            "fieldName": "EMAIL_MD5_HASH_",
            "fieldType": "STR50"
         },
         {
            "fieldName": "EMAIL_ISP_",
            "fieldType": "STR255"
         },
         {
            "fieldName": "EMAIL_FORMAT_",
            "fieldType": "CHAR"
         },
         {
            "fieldName": "EMAIL_DELIVERABILITY_STATUS_",
            "fieldType": "CHAR"
         },
         {
            "fieldName": "EMAIL_DOMAIN_",
            "fieldType": "STR255"
         }
      ],
      "links": [   {
         "rel": "createProfileExtensionTable",
         "href": "/rest/api/v1/lists/WS_Auto_RILIST/listExtensions",
         "method": "POST"
      }]
   }]
```


Sample Response in case of failure:


```
   {
      "type": "",
```

```
  "title": "List not found",
  "errorCode": "LIST_NOT_FOUND",
  "detail": "WS_Auto_RILISTs is an invalid list.",
  "errorDetails": []
}
```

Create a new profile extension table

A new profile extension table can be created for a given profile list by providing the schema of the profile extension table.

Service URL:

```
/rest/api/v1/lists/<listName>/listExtensions
```

Request Method:

```
POST
```

Request JSON Body:

```
{
        "profileExtension" : {
        "objectName":"ws_rest_petx",
        "folderName":"WS_REST_SAMPLE"},
        "fields":[{"fieldName":"edu", "fieldType" : "STR500"}]
}
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Response if successful:

```
true
```

Sample Response if failed:

```
{
        "type": "",
        "title": "Folder not found",
        "errorCode": "FOLDER_NOT_FOUND",
        "detail": "WS_REST_SAMPLESS Folder Not Found",
        "errorDetails": []
}
```

Merge or update members in a profile extension table

For an existing profile extension table, new members can be added or data for existing members can be updated. For a given profile extension table, an array of record data that contain field names and their corresponding field values are specified. Please see the definition of merge rule parameters provided at the end of this document. Up to 200 members can be processed in a single request.

Service URL:

```
/rest/api/v1/lists/<listName>/listExtensions/<petName>/members
```

Request Method:

```
POST
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request JSON Body:

```
{
          "recordData" : {
           "fieldNames" : ["riid_", "salary"],
           "records" : [
                          ["1761408", "10000"],
                          ["98798", "298909"],
                          ["xyz", "wedrfwe"],
                          ["12312", "23423","23423"],
                          ["1761409", "239482734"]
                       ],
                "mapTemplateName" : null
          },
          "insertOnNoMatch" : true,
          "updateOnMatch" : "REPLACE_ALL",
          "matchColumn" : "RIID"
}
```

Sample Response in case of success:

NOTE: Irrespective of what field names were used to perform the merge, the response will always contain only 'RIID_" in the 'fieldNames" attribute and the corresponding "RIID_" values for the records in the "records" attribute. In case merge failed for a record, the "RIID_" of the record is not present in the response. Instead, an error message starting with "MERGEFAILED:" is returned. Client developers can look for the string "MERGEFAILED:" in the response for a particular row to determine whether that recipient was merged successfully or not. Also note that the order of records in the response matches the order of records specified in the request payload. Furthermore, other attributes in the response like mapTemplateName, insertOnNoMatch, updateOnMatch and matchColumn will mirror the valid values specified in the request payload or default to null/false in case of invalid values.

MatchColumn attribute can have the following possible values: RIID, CUSTOMER_ID, EMAIL_ADDRESS, MOBILE_NUMBER, EMAIL_MD5_HASH and EMAIL_SHA256_HASH. Please note that new records will not be inserted for 'insertOnNoMatch' if matchColumn is either EMAIL_MD5_HASH or EMAIL_SHA256_HASH, even if matching records are not found in the table. The email hash attributes (EMAIL_MD5_HASH or EMAIL_SHA256_HASH) are only used for updating existing records.

```
{
   "recordData":    {
      "fieldNames": ["RIID_"],
      "records":       [
         ["1761408"],
         ["MERGEFAILED: Record 1 = RECORD DOES NOT MATCH ANY CONTACTS IN THE LIST
DemoNewsLetterList\r\n"],
         ["MERGEFAILED: Record 2 = ERROR: The value xyz is not valid for an
integer field\n\n\r\n"],
         ["MERGEFAILED: Record 3 = Field Names length, doesn't match with Field
Values length\r\n"],
         ["1761409"]
      ],
      "mapTemplateName": null
```

```
   },
   "insertOnNoMatch": true,
   "updateOnMatch": "REPLACE_ALL",
   "matchColumn": "RIID",
   "links":    [
      {
         "rel": "self",
         "href":
"/rest/api/v1/lists/DemoNewsLetterList/listExtensions/DemoNewsLetterPet/members",
         "method": "POST"
      },
      {
         "rel": "retrieveProfileExtensionRecipientsRIID",
         "href":
"/rest/api/v1/lists/DemoNewsLetterList/listExtensions/DemoNewsLetterPet/members/<
riid>",
         "method": "GET"
      },
      {
         "rel": "deleteProfileExtensionRecipientsRIID",
         "href":
"/rest/api/v1/lists/DemoNewsLetterList/listExtensions/DemoNewsLetterPet/members/<
riid>",
         "method": "DELETE"
      }
   ]
}
```


Sample Response in case of failure:


```
{
   "type": "",
   "title": "Invalid request parameters",
   "errorCode": "INVALID_PARAMETER",
   "detail": "Match Column is null",
   "errorDetails": []
}
```


Retrieve a member of a profile extension table based on RIID

Existing members of a profile extension table can be retrieved one at a time by specifying the member's Responsys ID (RIID). Please note that the total length of the string passed in for the fs parameter (containing the comma separated field names) cannot exceed 150 characters. To retrieve values of all columns, you can specify only one field with value set to 'all' (if you have a column called 'all', you should use two or more specific column names to avoid getting all of the columns).

Service URL:

```
/rest/api/v1/lists/<listName>/listExtensions/<petName>/members/<riid>
```

Request Method:

```
GET
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request parameters:

```
fs - comma separated list of fields to retrieve or 'all'
```

Sample Request:

```
/rest/api/v1/lists/DemoNewsLetterList/listExtensions/DemoNewsLetterPet/members/
1761408?fs=salary,riid_
```

Sample Response in case of success:

NOTE: Other attributes in the response like mapTemplateName, insertOnNoMatch, updateOnMatch and matchColumn will have default values (null/false).


```
{
   "recordData":    {
      "fieldNames":       [
         "SALARY",
         "RIID_"
      ],
      "records": [      [
         "10000",
         "1761408"
      ]],
      "mapTemplateName": null
   },
   "insertOnNoMatch": false,
   "updateOnMatch": null,
   "matchColumn": null,
   "links":    [
   {
         "rel": "self",
         "href":
"/rest/api/v1/lists/DemoNewsLetterList/listExtensions/DemoNewsLetterPet/members/1
761408?fs=salary,riid_",
         "method": "GET"
   },
   {
         "rel": "mergeProfileExtensionRecipients",
         "href":
"/rest/api/v1/lists/DemoNewsLetterList/listExtensions/DemoNewsLetterPet/members",
         "method": "POST"
   },
   {
         "rel": "deleteProfileExtensionRecipientsRIID",
         "href":
"/rest/api/v1/lists/DemoNewsLetterList/listExtensions/DemoNewsLetterPet/members/<
riid>",
         "method": "DELETE"
    }
   ]
}
```


Sample Response in case of failure:


```
{
   "type": "",
   "title": "Record not found",
   "errorCode": "RECORD_NOT_FOUND",
   "detail": "No records found in the table for the given ids",
   "errorDetails": []
}
```

Retrieve a member of a profile extension table based on a query attribute

Existing members of a profile extension table can be retrieved one at a time by specifying a unique identifier other than the member's Responsys ID (RIID) through the query attribute of the interface. Please note that the total length of the string passed in for the fs parameter (containing the comma separated field names) cannot exceed 150 characters. To retrieve values of all columns, you can specify only one field with value set to 'all' (if you have a column called 'all', you should use two or more specific column names to avoid getting all of the columns).

Service URL:

```
/rest/api/v1/lists/<listName>/listExtensions/<petName>/members
```

Request Method:

```
GET
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request Parameters:

```
qa – Query Attribute. Can be either 'r', 'e', 'c' or 'm'.
fs – Comma separated list of field names or 'all'
id – ID corresponding to the query attribute.
```

Sample Response in case of success:

NOTE: Other attributes in the response like mapTemplateName, insertOnNoMatch, updateOnMatch and matchColumn will have default values (null/false).


```
{
   "recordData":    {
      "fieldNames":       [
         "RIID_",
         "SALARY"
      ],
      "records": [      [
         "1761409",
         "239482734"
      ]],
      "mapTemplateName": null
   },
   "insertOnNoMatch": false,
   "updateOnMatch": null,
   "matchColumn": null,
   "links":    [
      {
         "rel": "self",
         "href":
"/rest/api/v1/lists/DemoNewsLetterList/listExtensions/DemoNewsLetterPet/members?q
a=e&fs=riid_,salary&id=responsysblr@gmail.com",
         "method": "GET"
      },
      {
         "rel": "mergeProfileExtensionRecipients",
```

```
            "href":
"/rest/api/v1/lists/DemoNewsLetterList/listExtensions/DemoNewsLetterPet/members",
            "method": "POST"
         },
         {
            "rel": "deleteProfileExtensionRecipientsRIID",
            "href":
"/rest/api/v1/lists/DemoNewsLetterList/listExtensions/DemoNewsLetterPet/members/<
riid>",
            "method": "DELETE"
         }
      ]
}
```

Sample Response in case of failure:

```
{
   "type": "",
   "title": "Record not found",
   "errorCode": "RECORD_NOT_FOUND",
   "detail": "No records found in the table for the given ids",
   "errorDetails": []
}
```

Delete a member of a profile extension table based on RIID

Existing members of a profile extension table can be deleted one at a time by specifying the Responsys ID (RIID).

Service URL:

```
/rest/api/v1/lists/<listName>/listExtensions/<petName>/members/<riid>
```

Request Method:

```
DELETE
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request parameters:

```
None.
```

Sample Response in case of success:

Note: The response will always contain only 'RIID_" in the 'fieldNames" attribute and the corresponding "RIID_" values for the records in the "records" attribute in case the deletion of that record is successful. In case delete failed for a record, the "RIID_" of the record is not present in the response. Instead, an error message starting with "DELETEFAILED:" is returned. Client developers can look for the string "DELETEFAILED:" in the response for a particular row to determine whether that recipient was deleted successfully or not. Furthermore, other attributes in the response like mapTemplateName, insertOnNoMatch, updateOnMatch and matchColumn will have default values (null/false).

```
{
   "recordData":    {
      "fieldNames": ["RIID_"],
      "records": [["1761409"]],
      "mapTemplateName": null
   },
   "insertOnNoMatch": false,
   "updateOnMatch": null,
```

```
   "matchColumn": null,
   "links":    [
      {
         "rel": "self",
         "href":
"/rest/api/v1/lists/DemoNewsLetterList/listExtensions/DemoNewsLetterPet/members/1
761409",
         "method": "DELETE"
      },
      {
         "rel": "mergeProfileExtensionRecipients",
         "href":
"/rest/api/v1/lists/DemoNewsLetterList/listExtensions/DemoNewsLetterPet/members",
         "method": "POST"
      },
      {
         "rel": "retrieveProfileExtensionRecipientsRIID",
         "href":
"/rest/api/v1/lists/DemoNewsLetterList/listExtensions/DemoNewsLetterPet/members/<
riid>",
         "method": "GET"
      }
   ]
}
```

Sample Response in case of failure:

```
{
   "recordData":    {
      "fieldNames": ["RIID_"],
      "records": [["DELETEFAILED: NO records found for id\n"]],
      "mapTemplateName": null
   },
   "insertOnNoMatch": false,
   "updateOnMatch": null,
   "matchColumn": null,
   "links":    [
      {
         "rel": "self",
         "href":
"/rest/api/v1/lists/DemoNewsLetterList/listExtensions/DemoNewsLetterPet/members/1
761409",
         "method": "DELETE"
      },
      {
         "rel": "mergeProfileExtensionRecipients",
         "href":
"/rest/api/v1/lists/DemoNewsLetterList/listExtensions/DemoNewsLetterPet/members",
         "method": "POST"
      },
      {
         "rel": "retrieveProfileExtensionRecipientsRIID",
         "href":
"/rest/api/v1/lists/DemoNewsLetterList/listExtensions/DemoNewsLetterPet/members/<
riid>",
         "method": "GET"
      }
   ]
}
```

Managing Supplemental Tables

New supplemental tables can be created, and for an existing supplemental table, its members can be added, updated, retrieved, or deleted.

Create a new supplemental table

A new supplemental table can be created by providing its schema.

Service URL:

```
/rest/api/v1/folders/<folderName>/suppData
```

Request Method:

```
POST
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request JSON Body:

```
{
"table" : {"objectName": "WS_REST_SUPPDATA_NEW"},
"fields" : [{
                "fieldName":"edu",
                "fieldType" : "STR500",
                "dataExtractionKey" : false},
                {
                "fieldName":"uni",
                "fieldType" : "STR500",
                "dataExtractionKey" : false
                },
                {
                "fieldName":"grade",
                "fieldType" : "STR500",
                "dataExtractionKey" : false
                }],
                "primaryKeys" : ["edu"]
}
```

Response if successful:

```
true
```

Sample Response if failed:

```
{
       "type": "",
               "title": "Invalid request parameters",
               "errorCode": "INVALID_PARAMETER",
               "detail": "Create Table Without PK is not supported.
               Please include Primary Key.",
               "errorDetails": []
}
```

Merge supplemental table records using primary key

For an existing supplemental table, new members can be added or data for existing members can be updated. For a given supplemental table in a specific folder, an array of record data that contains field names and their corresponding field values are specified along with ALL primary keys to identify the

desired record. Merging supplemental table records for a table without primary keys is not available. . A maximum of 200 records can be merged in one request.

Service URL:

```
/rest/api/v1/folders/<folderName>/suppData/<tableName>/members
```

Request Method:

```
POST
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request JSON body:

Note: All the Primary Key Fields of the table must be specified in the "fieldnames" attribute along with their corresponding values in the "records" attribute.

```
    {
  "recordData" : {
        "fieldNames" : ["PK1", "PK2", "F1", "F2", "PK3"],
        "records" : [
                    ["1", "1", "onerec", "onecol", "1"],
                    ["1", "1", "tworec", "twocol", "2"],
                    ["1", "2", "threerec", "threecol", "2"],
                    ["1", "2", "fourrec", "fourcol", "3"],
                    ["1", "1", "fiverec", "fivecol", "1", ""]
                    ],
        "mapTemplateName" : null
  },
   "insertOnNoMatch" : true,
   "updateOnMatch" : "REPLACE_ALL"
}
```

Sample Response in case of success:

NOTE: In case a record was merged successfully, the record in the response mirrors the record in the request payload. In case merge failed, the first element of the record contains an error message starting with "MERGEFAILED:" and depending on the number of elements present in the record in the request, the other elements in the record in the response are empty strings. Client developers can look for the string "MERGEFAILED:" in the response for a particular row to determine whether that recipient was merged successfully or not. Also note that the order of records in the response matches the order of records specified in the request payload. Also note that other attributes in the response like mapTemplateName, insertOnNoMatch, updateOnMatch and matchColumn will mirror the valid values specified in the request payload or default to null/false in case of invalid values.

```
{
   "recordData":    {
      "fieldNames":       [
         "PK1",
         "PK2",
         "F1",
         "F2",
         "PK3"
```

```
      ],
      "records":       [
         [
            "1",
            "1",
            "onerec",
            "onecol",
            "1"
         ],
         [
            "1",
            "1",
            "tworec",
            "twocol",
            "2"
         ],
         [
            "1",
            "2",
            "threerec",
            "threecol",
            "2"
         ],
         [
            "1",
            "2",
            "fourrec",
            "fourcol",
            "3"
         ],
         [
            "MERGEFAILED: Record 4 = Field Names length, doesn't match with Field
Values length\r\n",
            "",
            "",
            "",
            ""
         ]
      ],
      "mapTemplateName": null
   },
   "insertOnNoMatch": true,
   "updateOnMatch": "REPLACE_ALL",
   "links": [   {
      "rel": "self",
      "href":
"/rest/api/v1/folders/DemoNewsLetter/suppData/CompositePKSuppTable/members",
      "method": "POST"
   }]
}
```

Sample Response in case all primary key fields are not specified in the request payload:

```
{
   "recordData":    {
      "fieldNames":       [
         "PK1",
         "PK2",
         "F1",
         "F2"
      ],
      "records":       [
                  [
```

```
            "MERGEFAILED: Record 0 = NOT UPDATED PER MERGE RULE. MATCH FIELD
CANNOT BE EMPTY\r\n",
            "",
            "",
            ""
         ],
                  [
            "MERGEFAILED: Record 1 = NOT UPDATED PER MERGE RULE. MATCH FIELD
CANNOT BE EMPTY\r\n",
            "",
            "",
            ""
         ],
                  [
            "MERGEFAILED: Record 2 = NOT UPDATED PER MERGE RULE. MATCH FIELD
CANNOT BE EMPTY\r\n",
            "",
            "",
            ""
         ]
      ],
      "mapTemplateName": null
   },
   "insertOnNoMatch": true,
   "updateOnMatch": "REPLACE_ALL",
   "links": [   {
      "rel": "self",
      "href":
"/rest/api/v1/folders/DemoNewsLetter/suppData/CompositePKSuppTable/members",
      "method": "POST"
   }]
}
```

Sample Response in case of failure:

```
{
   "type": "",
   "title": "Invalid request parameters",
   "errorCode": "INVALID_PARAMETER",
   "detail": "Invalid template mapping xuy",
   "errorDetails": []
}
```

Merge supplemental table records without primary key

For an existing supplemental table, new members can be added or data for existing members can be updated. For a given supplemental table in a specific folder, an array of record data that contains field names and their corresponding field values are specified using a matchColumnNames attribute in the payload. A maximum of 200 records can be merged in one request.

Service URL:

```
/rest/api/v1/folders/<folderName>/suppData/<tableName>/members
```

Request Method:

```
POST
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request JSON body:

Note: the "matchColumnNames" attribute in the payload is used to identify the column to use to match to a record in the supplemental table without a primary key. Furthermore, this interface enforces that 'insertOnNoMatch' is 'true' and 'updateOnNoMatch' is 'REPLACE_ALL' in the request payload.

```
 {
"recordData" : {
       "fieldNames" : ["F1", "F2"],
       "records" : [
                   ["onerec", "updatedvalues"]
                   ],
       "mapTemplateName" : null
},
 "insertOnNoMatch" : true,
 "updateOnMatch" : "REPLACE_ALL",
 "matchColumnNames" : ["F1"]
}
```

Sample Response in case of success:

Note: In case a record was merged successfully, the record in the response mirrors the record in the request payload. In case merge failed for a record for some reason, the first element of the record contains an error message starting with "MERGEFAILED:" and depending on the number of elements present in the record in the request, the other elements in the record in the response are empty strings. Client developers can look for the string "MERGEFAILED:" in the response for a particular row to determine whether that recipient was merged successfully or not. Also note that the order of records in the response matches the order of records specified in the request payload. Furthermore, other attributes in the response (mapTemplateName, insertOnNoMatch, updateOnMatch and matchColumnNames) will mirror the valid values specified in the request payload or default to null/false in case of invalid values.

```
{
   "recordData":    {
      "fieldNames":       [
         "F1",
         "F2"
      ],
      "records": [[
         "onerec",
         "updatedvalues"
      ]],
      "mapTemplateName": null
   },
   "insertOnNoMatch": true,
   "updateOnMatch": "REPLACE_ALL",
   "matchColumnNames": ["F1"],
   "links": [   {
      "rel": "self",
      "href":
"/rest/api/v1/folders/DemoNewsLetter/suppData/CompositePKSuppTable/members",
      "method": "POST"
   }]
}
```

Sample Response in case matchColumnNames and fieldNames specified cannot be used to identify records to merge:

Sample Payload:

```
{
 "recordData" : {
        "fieldNames" : ["PK1", "PK2", "F1", "F2", "PK3"],
        "records" : [
                    ["1", "1", "onerec", "onecol", "1"],
                    ["1", "1", "onerec", "onecol", "2"],
                    ["1", "1", "onerec", "onecol", "3"]
                    ],
        "mapTemplateName" : null
 },
  "insertOnNoMatch" : true,
  "updateOnMatch" : "REPLACE_ALL",
  "matchColumnNames" : ["F1"]
}
```

Response:

```
 {
  "recordData":    {
     "fieldNames":       [
        "PK1",
        "PK2",
        "F1",
        "F2",
        "PK3"
     ],
     "records":       [
                 [
           "MERGEFAILED: Unable to identify record to Merge from the Match
Columns and FieldNames Specified.",
           "",
           "",
           "",
           ""
      ],
                 [
           "MERGEFAILED: Unable to identify record to Merge from the Match
Columns and FieldNames Specified.",
           "",
           "",
           "",
           ""
      ],
                 [
           "MERGEFAILED: Unable to identify record to Merge from the Match
Columns and FieldNames Specified.",
           "",
           "",
           "",
           ""
      ]
     ],
     "mapTemplateName": null
   },
   "insertOnNoMatch": true,
   "updateOnMatch": "REPLACE_ALL",
   "matchColumnNames": ["F1"],
   "links": [   {
```

```
            "rel": "self",
            "href":
      "/rest/api/v1/folders/DemoNewsLetter/suppData/CompositePKSuppTable/members",
            "method": "POST"
          }]
      }
```

Sample Response in case insertOnMatch or updateOnNoMatch are invalid

Sample Payload:

```
      {
       "recordData" : {
              "fieldNames" : ["F1", "F2"],
              "records" : [
                          ["onerec", "updatedvalues"]
                          ],
              "mapTemplateName" : null
        },
         "insertOnNoMatch" : false,
         "updateOnMatch" : "REPLACE_ALL",
         "matchColumnNames" : ["F1"]
      }
```

Response:

```
  {
   "type": "",
   "title": "Invalid request parameters",
   "errorCode": "INVALID_PARAMETER",
   "detail": "insertOnNoMatch must be true and updateOnMatch must be REPLACE_ALL if
matchColumnNames is specified",
   "errorDetails": []
  }
```

Sample Response in case matchColumnNames is not specified:

Sample Payload:

```
      {
       "recordData" : {
              "fieldNames" : ["F1", "F2"],
              "records" : [
                          ["onerec", "updatedvalues"]
                          ],
              "mapTemplateName" : null
        },
         "insertOnNoMatch" : false,
         "updateOnMatch" : "REPLACE_ALL",
         "matchColumnNames" : null
      }
```

Response:

```
  {
   "type": "",
   "title": "Invalid request parameters",
   "errorCode": "INVALID_PARAMETER",
   "detail": "MatchColumnNames must be specified to merge into a table that does not
have any primary keys",
   "errorDetails": []
```

```
}
```

Retrieve supplemental table records with primary key

This interface is to retrieve Supplemental Table Records by specifying the primary key values using the request parameters. Please note that the total length of the string passed in for the fs parameter (containing the comma separated field names) cannot exceed 150 characters. To retrieve values of all columns, you can specify only one field with value set to 'all' (if you have a column called 'all', you should use two or more specific column names to avoid getting all of the columns).

Service URL:

```
/rest/api/v1 /folders/<folderName>/suppData/<tableName>/members
```

Request Method:

```
GET
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request Parameters:

```
qa – Query Attribute. All of the Primary Key values of the Supplemental Table
     must be specified by repeating this parameter.
fs – Comma separated list of field names or 'all'
id – IDs corresponding to the query attribute. All the Primary Key Values of
     the Supplemental Table must be specified by repeating this parameter. The
     order of the values must match the order of the Primary Keys specified in
     the 'qa' parameter.
```

Sample Request:

```
/rest/api/v1/folders/DemoNewsLetter/suppData/CompositePKSuppTable/members?qa=PK
1&qa=PK2&qa=PK3&fs=PK1,PK2,PK3,F1,F2&id=1&id=1&id=1
```

Sample Response in case of success:

NOTE: Other attributes in the response like mapTemplateName, insertOnNoMatch, updateOnMatch and matchColumn will have default values (null/false).

```
{
    "recordData":{
        "fieldNames":[
            "PK1",
            "PK2",
            "PK3",
            "F1",
            "F2"
        ],
        "records":[
        [
                "1",
                "1",
                "1",
                "onerec",
                "onecol"
         ]
        ],
```

```
            "mapTemplateName":null
        },
        "insertOnNoMatch":false,
        "updateOnMatch":null,
        "links":[
            {
                "rel":"self",

    "href":"/rest/api/v1/folders/DemoNewsLetter/suppData/CompositePKSuppTable/members
    ?qa=PK1&qa=PK2&qa=PK3&fs=PK1,PK2,PK3,F1,F2&id=1&id=1&id=1",
                "method":"GET"
            },
            {
                "rel":"mergeTableMembers",

    "href":"/rest/api/v1/folders/DemoNewsLetter/suppData/CompositePKSuppTable/members
    ",
                "method":"POST"
            }
        ]
    }
```


Sample Response in case of failure:


```
    {
       "type": "",
       "title": "Invalid field name",
       "errorCode": "INVALID_FIELD_NAME",
       "detail": "Column(s) [F1] is not indexed",
       "errorDetails": []
    }
```


Sample Response in case ALL Primary Key Columns are not specified in the 'qa' request parameter:


```
    {
       "type": "",
       "title": "Invalid request parameters",
       "errorCode": "INVALID_PARAMETER",
       "detail": "All and Only the Primary Keys in the Table [PK1, PK2, PK3] must be
    specified as Query Columns.",
       "errorDetails": []
    }
```


Delete supplemental table records

This interface is to delete a Supplemental Table Records by specifying the primary key using request parameters.

Service URL:

```
/rest/api/v1/lists /folders/<folderName>/suppData/<tableName>/members
```

Request Method:

```
DELETE
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request Parameters:

```
qa – Query Attribute. All the Primary Keys of the Supplemental Table must be
     specified by repeating this parameter.
id – IDs corresponding to the query attribute. All the Primary Key Values of
     the Supplemental Table must be specified by repeating this parameter. The
     order of the values must match the order of the Primary Keys specified in
     the 'qa' parameter.
```

Sample Request:

```
rest/api/v1/folders/DemoNewsLetter/suppData/CompositePKSuppTable/members?qa=PK1
&qa=PK2&qa=PK3&fs=PK1,PK2,PK3,F1,F2&id=1&id=1&id=1
```

Sample Response in case of success:

NOTE: The response will always contain all the query attributes specified in the request parameter in the 'fieldNames" attribute and the corresponding values for the records in the "records" attribute in case the deletion of that record is successful. In case a record is not found for the given id, an error response saying "No Records found for the Id" is returned. In case a record is found and delete failed for some reason, the record in the response contains an error message starting with "DELETEFAILED:". Client developers can look for the string "DELETEFAILED:" in the response for a particular row to determine whether that record was deleted successfully or not. Furthermore, other attributes in the response like mapTemplateName, insertOnNoMatch, updateOnMatch and matchColumn will have default values (null/false).


```
{
    "recordData":{
        "fieldNames":[
            "PK1",
            "PK2",
            "PK3"
        ],
        "records":[
            [
                "1",
                "1",
                "1"
            ]
        ],
        "mapTemplateName":null
    },
    "insertOnNoMatch":false,
    "updateOnMatch":null,
    "links":[
        {
            "rel":"self",

"href":"/rest/api/v1/folders/DemoNewsLetter/suppData/CompositePKSuppTable/members
?qa=PK1&qa=PK2&qa=PK3&fs=PK1,PK2,PK3,F1,F2&id=1&id=1&id=1",
            "method":"DELETE"
        },
        {
            "rel":"mergeTableMembers",

"href":"/rest/api/v1/folders/DemoNewsLetter/suppData/CompositePKSuppTable/members
",
            "method":"POST"
        }
```

```
    ]
}
```

Sample Response in case of failure:

```
{
   "type": "",
   "title": "Record not found",
   "errorCode": "RECORD_NOT_FOUND",
   "detail": "No records found in the table for the given ids",
   "errorDetails": []
}
```

Sample Response in case ALL Primary Key Columns are not specified in the 'qa' request parameter:

```
{
   "type": "",
   "title": "Invalid request parameters",
   "errorCode": "INVALID_PARAMETER",
   "detail": "All and Only the Primary Keys in the Table [PK1, PK2, PK3] must be
specified as Query Columns.",
   "errorDetails": []
}
```

Triggering Email Messages

Responsys email campaigns that already exist can be sent to up to 200 members of a profile list. The following allows merging members into a profile list and subsequently sending them an email message. For the list associated with the specified email campaign, an array of record data that contain field names and their corresponding field values are specified. Please see the definition of merge rule parameters provided at the end of this document.

Merge members into a profile list and trigger email messages to them

Service URL:

```
/rest/api/v1/campaigns/<campaignName>/email
```

Request Method:

```
POST
```

Request JSON Body:

```
{
"recordData": {
   "records": [
       {
           "fieldValues": [
               "mdi1234@foobar.com",
               "martiness"
           ]
       },
       {
           "fieldValues": [
               "mdi.1234@foobarcorp.com",
               "concord"
           ]
       }
   ],
```

```
            "fieldNames": [
                "EMAIL_ADDRESS_",
                "CITY_"
            ]
        },
        "mergeRule": {
            "htmlValue": "H",
            "matchColumnName1": "EMAIL_ADDRESS_",
            "matchColumnName2": null,
            "matchColumnName3": null,
            "optoutValue": "O",
            "insertOnNoMatch": true,
            "defaultPermissionStatus": "OPTIN",
            "rejectRecordIfChannelEmpty": "E",
            "optinValue": "I",
            "updateOnMatch": "REPLACE_ALL",
            "textValue": "T",
            "matchOperator": "NONE"
        },
        "triggerData": [
            {
                "optionalData": [
                    {
                        "name": "FIRST_NAME",
                        "value": "jim_1"
                    },
                    {
                        "name": "LAST_NAME",
                        "value": "smith_1"
                    }
                ]
            },
            {
                "optionalData": [
                    {
                        "name": "FIRST_NAME",
                        "value": "jim_2"
                    },
                    {
                        "name": "LAST_NAME",
                        "value": "smith_2"
                    }
                ]
            }
        ]
}
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Sample Response:

```
[
 {
  "errorMessage" : null,
  "success" : true,
  "recipientId" : 72067
 },
 {
  "errorMessage" : null,
  "success" : true,
  "recipientId" : 72087
 }
```

```
]
```

Trigger email message

The following allows triggering email messages to existing members of a profile list.

Service URL:

```
/rest/api/v1/campaigns/<campaignName>/email
```

Request Method:

```
POST
```

Request JSON Body:

```
{
  "recipientData" : [{
   "recipient" : {
       "customerId" : "1",
       "emailAddress" : "foo.bar@oracle.com",
       "listName" : {
        "folderName" : "WS_REST_SAMPLE",
        "objectName" : "wsrest"
       },
       "recipientId" : null,
       "mobileNumber" : null,
       "emailFormat" : "HTML_FORMAT"
   },
   "optionalData" : [{
        "name" : "CUSTOM1",
        "value" : "c1a_value_new"
       }, {
        "name" : "CUSTOM2",
        "value" : "c2a_value_new"
       }
   ]
  }, {
   "recipient" : {
       "customerId" : "2",
       "emailAddress" : "baz.foo@oracle.com",
       "listName" : {
        "folderName" : "WS_REST_SAMPLE",
        "objectName" : "wsrest"
       },
       "recipientId" : null,
       "mobileNumber" : null,
       "emailFormat" : "TEXT_FORMAT"
   },
   "optionalData" : [{
        "name" : "CUSTOM1",
        "value" : "c1b_value_new"
       }, {
        "name" : "CUSTOM2",
        "value" : "c2b_value_new"
       }
   ]
  }
 ]
}
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Sample Response:

```
[{
  "errorMessage" : null,
  "success" : true,
  "recipientId" : 72067
 }, {
  "errorMessage" : "NO_RECIPIENT_FOUND",
  "success" : false,
  "recipientId" : -1
 }
]
```

Triggering SMS Messages

Responsys SMS campaigns that already exist can be sent to up to 200 members of a profile list. The following allows merging members into a profile list and subsequently sending them an SMS message. For the list associated with the specified SMS campaign, an array of record data that contain field names and their corresponding field values are specified. Please see the definition of merge rule parameters provided at the end of this document.

Merge members into a profile list and trigger SMS messages to them

Service URL:

```
/rest/api/v1/campaigns/<campaignName>/sms
```

Request Method:

```
POST
```

Request JSON Body:

```
{
 "recordData" : {
  "fieldNames" : ["CUSTOMER_ID_", "EMAIL_ADDRESS_", "MOBILE_NUMBER_",
"MOBILE_COUNTRY_"],
  "records" : [{
      "fieldValues" : ["1001", "foo.bar@oracle.com", "6505551212", "US"]
   }, {
      "fieldValues" : ["1002", "baz.foo@oracle.com", "6505551212", "US"]
   }
  ]
 },

 "mergeRule" : {
  "insertOnNoMatch" : true,
  "updateOnMatch" : "NO_UPDATE",
  "matchColumnName1" : "CUSTOMER_ID_",
  "matchColumnName2" : null,
  "matchOperator" : "NONE",
  "optinValue" : "I",
  "optoutValue" : "O",
  "htmlValue" : "H",
  "textValue" : "T",
  "rejectRecordIfChannelEmpty" : "E",
  "defaultPermissionStatus" : "OPTIN"
 },
 "triggerData" : [{
   "optionalData" : [{
      "name" : "CITY_",
      "value" : "San Bruno"
```

```
        }
      ]
     }
    ]
   }
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Sample Response:

```
[{
  "errorMessage" : null,
  "success" : true,
  "recipientId" : 72067
 }, {
  "errorMessage" : null,
  "success" : true,
  "recipientId" : 72087
 }
]
```

Raising Custom Events for Cross-channel Marketing Programs

You can set up a Program to listen for one or more custom events, which in turn can start a program or be used in a program event switch. The following API is used to trigger a specific custom event. The Program will use the existing members of a profile list that are specified in the API request.

Trigger a custom event

Service URL:

```
/rest/api/v1/events/<eventName>
```

Request Method:

```
POST
```

Request JSON Body:

```
{
 "customEvent" : {
  "eventNumberDataMapping" : null,
  "eventDateDataMapping" : null,
  "eventStringDataMapping" : null
 },
 "recipientData" : [{
   "recipient" : {
       "customerId" : 1,
       "emailAddress" : null,
       "listName" : {
        "folderName" : "WS_REST_SAMPLE",
        "objectName" : "wsrest"
       },
       "recipientId" : null,
       "mobileNumber" : null,
       "emailFormat" : "HTML_FORMAT"
```

```
    },
    "optionalData" : [{
         "name" : "CUSTOM1",
         "value" : "value1"
        }
    ]
   }
  ]
}
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Sample Response:

```
[{
  "errorMessage" : null,
  "success" : true,
  "recipientId" : 72067
 }
]
```

Managing Email Campaigns

The following interfaces are for creating and manipulating Responsys EMD email campaign objects.

Create email campaign

This API is for creating an EMD email campaign object.

Service URL:

```
/rest/api/v1/campaigns
```

Request Method:

```
POST
```

Required Attributes:

```
Name
folderName
```

Request JSON Body:

```
{
    "name": "testcampaign-b11",
    "folderName": "testfolder",
    "type": "EMAIL",
    "description": "<description>",
    "purpose": "PROMOTIONAL",
    "marketingStrategy": "<strategy>",
    "marketingProgram": "<program>",
    "listName": "<listname>",
    "filterPaths": [
        "foldername/objectName1",
        "foldername/objectName2"
    ],
    "refiningDataSourcePath": "foldername/objectName1",
    "proofListPath": "foldername/objectName1",
    "seedListPath": "foldername/objectName1",
```

```
        "segmentPaths": [
            "foldername/objectName1",
            "foldername/objectName2"
        ],
        "supplementaryCampaignDataSourcePaths": [
            "foldername/objectName1",
            "foldername/objectName2"
        ],
        "supplementaryProofDataSourcePaths": [
            "foldername/objectName1",
            "foldername/objectName2"
        ],
        "supplementarySeedDataSourcePaths": [
            "foldername/objectName1",
            "foldername/objectName2"
        ],
        "suppressionListPaths": [
            "foldername/objectName1",
            "foldername/objectName2"
        ],
        "subject": "<subject>",
        "fromName": "<from name>",
        "fromEmail": "<from email>",
        "replyToEmail": "<reply to email>",
        "bccEmail": "<bcc email>",
        "htmlMessagePath": "documentPath",
        "textMessagePath": "documentPath",
        "enableExternalTracking": true,
        "externalTrackingParams": {
            "name1": "value1",
            "name2": "value2"
        },
        "enableLinkTracking": true,
        "linkTablePath": "foldername/objectName1",
        "attachmentPaths": [
            "documentPath1",
            "documentPath2"
        ],
        "campaignVariables": {
            "name1": "value1",
            "name2": "value2"
        },
        "useUTF8": true,
        "locale": "<value>",
        "trackHTMLOpens": true,
        "trackConversions": true,
        "sendTextIfHTMLUnknown": true,
        "segmentTrackingColumnName": "<name>",
        "unsubscribeOption": "OPTOUT_SINGLE_CLICK",
        "unsubscribeFormName": "name",
        "autoCloseOption": "NO_AUTO_CLOSE",
        "autoCloseValue": "<value>",
        "closedCampaignURL": "<URL>",
        "externalCampaignCode": "<code>",
        "salesForceCampaignId": "<salesforce id>"
    }
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Response:

```
{
    "id": 1000,
    "name": "testcampaign-b11",
    "folderName": "testfolder",
    "type": "EMAIL",
    "description": "<description>",
    "purpose": "PROMOTIONAL",
    "marketingStrategy": "<strategy>",
    "marketingProgram": "<program>",
    "listName": "<listname>",
    "filterPaths": [
        "foldername/objectName1",
        "foldername/objectName2"
    ],
    "refiningDataSourcePath": "foldername/objectName1",
    "proofListPath": "foldername/objectName1",
    "seedListPath": "foldername/objectName1",
    "segmentPaths": [
        "foldername/objectName1",
        "foldername/objectName2"
    ],
    "supplementaryCampaignDataSourcePaths": [
        "foldername/objectName1",
        "foldername/objectName2"
    ],
    "supplementaryProofDataSourcePaths": [
        "foldername/objectName1",
        "foldername/objectName2"
    ],
    "supplementarySeedDataSourcePaths": [
        "foldername/objectName1",
        "foldername/objectName2"
    ],
    "suppressionListPaths": [
        "foldername/objectName1",
        "foldername/objectName2"
    ],
    "subject": "<subject>",
    "fromName": "<from name>",
    "fromEmail": "<from email>",
    "replyToEmail": "<reply to email>",
    "bccEmail": "<bcc email>",
    "htmlMessagePath": "documentPath",
    "textMessagePath": "documentPath",
    "enableExternalTracking": true,
    "externalTrackingParams": {
        "name1": "value1",
        "name2": "value2"
    },
    "enableLinkTracking": true,
    "linkTablePath": "foldername/objectName1",
    "attachmentPaths": [
        "documentPath1",
        "documentPath2"
    ],
    "campaignVariables": {
        "name1": "value1",
        "name2": "value2"
    },
    "useUTF8": true,
    "locale": "<value>",
    "trackHTMLOpens": true,
    "trackConversions": true,
```

```
    "sendTextIfHTMLUnknown": true,
    "segmentTrackingColumnName": "<name>",
    "unsubscribeOption": "OPTOUT_SINGLE_CLICK",
    "unsubscribeFormName": "name",
    "autoCloseOption": "NO_AUTO_CLOSE",
    "autoCloseValue": "<value>",
    "closedCampaignURL": "<URL>",
    "externalCampaignCode": "<code>",
    "salesForceCampaignId": "<salesforce id>",
    "links": [
        {
            "rel": "self",
            "href": "/rest/api/v1/campaigns",
            "method": "POST"
        },
        {
            "rel": "getCampaign",
            "href": "/rest/api/v1/campaigns/<campaign_name>",
            "method": "GET"
        },
        {
            "rel": "updateCampaign",
            "href": "rest/api/v1/campaigns/<campaign_name>",
            "method": "PUT"
        }
    ]
}
```

Update email campaign

This API is for updating an existing EMD email campaign object.

Service URL:

```
/rest/api/v1/campaigns/<campaign_name>
```

Request Method:

```
PUT
```

Required Attributes:

```
None
```

Request JSON Body:

```
{
    "description": "<description>",
    "purpose": "PROMOTIONAL",
    "marketingStrategy": "<strategy>",
    "marketingProgram": "<program>",
    "listName": "<listname>",
    "filterPaths": [
        "foldername/objectName1",
        "foldername/objectName2"
    ],
    "refiningDataSourcePath": "foldername/objectName1",
    "proofListPath": "foldername/objectName1",
    "seedListPath": "foldername/objectName1",
    "segmentPaths": [
        "foldername/objectName1",
        "foldername/objectName2"
    ],
    "supplementaryCampaignDataSourcePaths": [
```

```
            "foldername/objectName1",
            "foldername/objectName2"
        ],
        "supplementaryProofDataSourcePaths": [
            "foldername/objectName1",
            "foldername/objectName2"
        ],
        "supplementarySeedDataSourcePaths": [
            "foldername/objectName1",
            "foldername/objectName2"
        ],
        "suppressionListPaths": [
            "foldername/objectName1",
            "foldername/objectName2"
        ],
        "subject": "<subject>",
        "fromName": "<from name>",
        "fromEmail": "<from email>",
        "replyToEmail": "<reply to email>",
        "bccEmail": "<bcc email>",
        "htmlMessagePath": "documentPath",
        "textMessagePath": "documentPath",
        "enableExternalTracking": true,
        "externalTrackingParams": {
            "name1": "value1",
            "name2": "value2"
        },
        "enableLinkTracking": true,
        "linkTablePath": "foldername/objectName1",
        "attachmentPaths": [
            "documentPath1",
            "documentPath2"
        ],
        "campaignVariables": {
            "name1": "value1",
            "name2": "value2"
        },
        "useUTF8": true,
        "locale": "<value>",
        "trackHTMLOpens": true,
        "trackConversions": true,
        "sendTextIfHTMLUnknown": true,
        "segmentTrackingColumnName": "<name>",
        "unsubscribeOption": "OPTOUT_SINGLE_CLICK",
        "unsubscribeFormName": "name",
        "autoCloseOption": "NO_AUTO_CLOSE",
        "autoCloseValue": "<value>",
        "closedCampaignURL": "<URL>",
        "externalCampaignCode": "<code>",
        "salesForceCampaignId": "<salesforce id>"
    }
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Response:

```
{
    "id": 1000,
    "name": "testcampaign-b11",
    "folderName": "testfolder",
    "type": "EMAIL",
    "description": "<description>",
    "purpose": "PROMOTIONAL",
```

```
"marketingStrategy": "<strategy>",
"marketingProgram": "<program>",
"listName": "<listname>",
"filterPaths": [
    "foldername/objectName1",
    "foldername/objectName2"
],
"refiningDataSourcePath": "foldername/objectName1",
"proofListPath": "foldername/objectName1",
"seedListPath": "foldername/objectName1",
"segmentPaths": [
    "foldername/objectName1",
    "foldername/objectName2"
],
"supplementaryCampaignDataSourcePaths": [
    "foldername/objectName1",
    "foldername/objectName2"
],
"supplementaryProofDataSourcePaths": [
    "foldername/objectName1",
    "foldername/objectName2"
],
"supplementarySeedDataSourcePaths": [
    "foldername/objectName1",
    "foldername/objectName2"
],
"suppressionListPaths": [
    "foldername/objectName1",
    "foldername/objectName2"
],
"subject": "<subject>",
"fromName": "<from name>",
"fromEmail": "<from email>",
"replyToEmail": "<reply to email>",
"bccEmail": "<bcc email>",
"htmlMessagePath": "documentPath",
"textMessagePath": "documentPath",
"enableExternalTracking": true,
"externalTrackingParams": {
    "name1": "value1",
    "name2": "value2"
},
"enableLinkTracking": true,
"linkTablePath": "foldername/objectName1",
"attachmentPaths": [
    "documentPath1",
    "documentPath2"
],
"campaignVariables": {
    "name1": "value1",
    "name2": "value2"
},
"useUTF8": true,
"locale": "<value>",
"trackHTMLOpens": true,
"trackConversions": true,
"sendTextIfHTMLUnknown": true,
"segmentTrackingColumnName": "<name>",
"unsubscribeOption": "OPTOUT_SINGLE_CLICK",
"unsubscribeFormName": "name",
"autoCloseOption": "NO_AUTO_CLOSE",
"autoCloseValue": "<value>",
"closedCampaignURL": "<URL>",
```

```
    "externalCampaignCode": "<code>",
    "salesForceCampaignId": "<salesforce id>",
    "links": [
        {
            "rel": "self",
            "href": "/rest/api/v1/campaigns/<campaign_name>",
            "method": "PUT"
        },
        {
            "rel": "create",
            "href": "/rest/api/v1/campaigns",
            "method": "POST"
        },
        {
            "rel": "getProperties",
            "href": "/rest/api/v1/campaigns/<campaign_name>",
            "method": "GET"
        }
    ]
}
```

Get email campaign

This API is for getting an existing EMD email campaign object.

Service URL:

```
/rest/api/v1/campaigns/<campaign_name>
```

Request Method:

```
GET
```

Required Attributes:

```
None
```

Request JSON Body:

```
None
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Response:

```
  {
    "id": 1000,
    "name": "testcampaign-b11",
    "folderName": "testfolder",
    "type": "EMAIL",
    "description": "<description>",
    "purpose": "PROMOTIONAL",
    "marketingStrategy": "<strategy>",
    "marketingProgram": "<program>",
    "listName": "<listname>",
    "filterPaths": [
        "foldername/objectName1",
        "foldername/objectName2"
    ],
    "refiningDataSourcePath": "foldername/objectName1",
    "proofListPath": "foldername/objectName1",
    "seedListPath": "foldername/objectName1",
    "segmentPaths": [
```

```
        "foldername/objectName1",
        "foldername/objectName2"
    ],
    "supplementaryCampaignDataSourcePaths": [
        "foldername/objectName1",
        "foldername/objectName2"
    ],
    "supplementaryProofDataSourcePaths": [
        "foldername/objectName1",
        "foldername/objectName2"
    ],
    "supplementarySeedDataSourcePaths": [
        "foldername/objectName1",
        "foldername/objectName2"
    ],
    "suppressionListPaths": [
        "foldername/objectName1",
        "foldername/objectName2"
    ],
    "subject": "<subject>",
    "fromName": "<from name>",
    "fromEmail": "<from email>",
    "replyToEmail": "<reply to email>",
    "bccEmail": "<bcc email>",
    "htmlMessagePath": "documentPath",
    "textMessagePath": "documentPath",
    "enableExternalTracking": true,
    "externalTrackingParams": {
        "name1": "value1",
        "name2": "value2"
    },
    "enableLinkTracking": true,
    "linkTablePath": "foldername/objectName1",
    "attachmentPaths": [
        "documentPath1",
        "documentPath2"
    ],
    "campaignVariables": {
        "name1": "value1",
        "name2": "value2"
    },
    "useUTF8": true,
    "locale": "<value>",
    "trackHTMLOpens": true,
    "trackConversions": true,
    "sendTextIfHTMLUnknown": true,
    "segmentTrackingColumnName": "<name>",
    "unsubscribeOption": "OPTOUT_SINGLE_CLICK",
    "unsubscribeFormName": "name",
    "autoCloseOption": "NO_AUTO_CLOSE",
    "autoCloseValue": "<value>",
    "closedCampaignURL": "<URL>",
    "externalCampaignCode": "<code>",
    "salesForceCampaignId": "<salesforce id>",
    "links": [
        {
            "rel": "self",
            "href": "/rest/api/v1/campaigns/<campaign_name>",
            "method": "GET"
        },
        {
            "rel": "create",
            "href": "/rest/api/v1/campaigns",
```

```
            "method": "POST"
        },
        {
            "rel": "updateCampaign",
            "href": "rest/api/v1/campaigns/<campaign_name>",
            "method": "PUT"
        }
    ]
}
```

Schedule email campaign

This API is for scheduling an existing EMD email campaign for immediate or future launch.

Service URL:

```
/rest/api/v1/campaigns/{campaignName}/schedule
```

Request Method:

```
POST
```

Required Attributes:

```
scheduleType (ONCE or NOW)
```

Request JSON Body:

```
{
    "scheduleType": "ONCE",
    "scheduledTime": "2015-05-25 06:00 AM",
    "launchOptions": {
        "proofLaunch": true,
        "proofLaunchEmail": "someemail@a.com",
        "proofLaunchType": "LAUNCH_TO_EMAIL",
         "recipientLimit": 3,
        "samplingNthSelection": 1,
        "samplingNthOffset": 1,
        "samplingNthInterval": 1,
        "progressEmailAddresses": [
            "email1@a.com",
            "email2@a.com"
        ],
        "progressChunk": "CHUNK_10K"
    }
}
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Response:

```
{
    "id": 1,
    "scheduleType": "ONCE",
    "scheduledTime": "2015-05-25 06:00 AM",
    "launchOptions": {
        "proofLaunch": true,
        "proofLaunchEmail": "someemail@a.com",
        "proofLaunchType": "LAUNCH_TO_EMAIL",
        "recipientLimit": 3,
        "samplingNthSelection": 1,
        "samplingNthOffset": 1,
```

```
        "samplingNthInterval": 1,
        "progressEmailAddresses": [
            "email1@a.com",
            "email2@a.com"
        ],
        "progressChunk": "CHUNK_10K",
        "links": [
            {
                "rel": "self",
                "href": "/rest/api/v1/campaigns/<campaignName>/schedule",
                "method": "POST"
            },
            {
                "rel": "getSchedule",
                "href": "/rest/api/v1/campaigns/<campaign_name>/schedule/<id>",
                "method": "GET"
            },
            {
                "rel": "updateSchedule",
                "href": "rest/api/v1/campaigns/<campaign_name>/schedule/<id>",
                "method": "PUT"
            },
            {
                "rel": "deleteSchedule",
                "href": "rest/api/v1/campaigns/<campaign_name>/schedule/<id>",
                "method": "DELETE"
            }
        ]
    }
```

Get email campaign schedule

This API is for getting the schedule of an EMD campaign using the campaign schedule ID that was returned from the schedule campaign API.

Service URL:

```
/rest/api/v1/campaigns/{campaignName}/schedule/{scheduleId}
```

Request Method:

```
GET
```

Required Attributes:

```
campaignName
scheduleId
```

Request:

```
None
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Response:

```
{
    "id": 1,
    "scheduleType": "ONCE",
    "scheduledTime": "2015-05-25 06:00 AM",
    "launchOptions": {
        "proofLaunch": true,
```

```
        "proofLaunchEmail": "someemail@a.com",
        "proofLaunchType": "LAUNCH_TO_EMAIL",
        "recipientLimit": 3,
        "samplingNthSelection": 1,
        "samplingNthOffset": 1,
        "samplingNthInterval": 1,
        "progressEmailAddresses": [
            "email1@a.com",
            "email2@a.com"
        ],
        "progressChunk": "CHUNK_10K",
        "links": [
            {
                "rel": "self",
                "href": "/rest/api/v1/campaigns/<campaignName>/schedule/<scheduleId>",
                "method": "POST"
            },
            {
                "rel": "createSchedule",
                "href": "/rest/api/v1/campaigns/<campaign_name>/schedule",
                "method": "GET"
            },
            {
                "rel": "updateSchedule",
                "href": "rest/api/v1/campaigns/<campaign_name>/schedule/<id>",
                "method": "PUT"
            },
            {
                "rel": "deleteSchedule",
                "href": "rest/api/v1/campaigns/<campaign_name>/schedule/<id>",
                "method": "DELETE"
            }
        ]
    }
}
```

Update email campaign schedule

This API is for updating the schedule of an existing EMD email campaign using the schedule ID that was returned from schedule campaign API.

Service URL:

```
/rest/api/v1/campaigns/{campaignName}/schedule/{scheduleID}
```

Request Method:

```
PUT
```

Required Attributes:

```
campaignName
scheduleId
```

Request:

```
{
   "scheduleType": "ONCE",
   "scheduledTime": "2015-11-30 1:00 AM"
}
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Response:


```
{
  "id": 1491,
  "scheduleType": "ONCE",
  "scheduledTime": "2015-11-30 01:00 AM",
  "launchOptions": {
    "proofLaunch": false
  },
  "links": [
    {
      "rel": "self",
      "href": "/rest/api/v1/campaigns/test/schedule/1491",
      "method": "PUT"
    },
    {
      "rel": "deleteSchedule",
      "href": "/rest/api/v1/campaigns/test/schedule/1491",
      "method": "DELETE"
    },
    {
      "rel": "getSchedule",
      "href": "/rest/api/v1/campaigns/test/schedule/1491",
      "method": "GET"
    },
    {
      "rel": "createSchedule",
      "href": "/rest/api/v1/campaigns/test/schedule",
      "method": "POST"
    }
  ]
}
```


Delete email campaign schedule

This API is for deleting the schedule of an existing EMD email campaign by using the schedule ID returned from the campaign schedule.

Service URL:

```
/rest/api/v1/campaigns/{campaignName}/schedule/{scheduleId}
```

Request Method:

```
DELETE
```

Required Attributes:

```
campaignName
scheduleId
```

Request:

```
None
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Response:


```
{
"id": 1491,
  "scheduleType": "ONCE",
  "scheduledTime": "2015-11-30 01:00 AM",
```

```
    "launchOptions": {
      "proofLaunch": false
    },
    "links": [
      {
        "rel": "self",
        "href": "/rest/api/v1/campaigns/test/schedule/1491",
        "method": "DELETE"
      },
      {
        "rel": "updateSchedule",
        "href": "/rest/api/v1/campaigns/test/schedule/1491",
        "method": "PUT"
      },
      {
        "rel": "getSchedule",
        "href": "/rest/api/v1/campaigns/test/schedule/1491",
        "method": "GET"
      },
      {
        "rel": "createSchedule",
        "href": "/rest/api/v1/campaigns/test/schedule",
        "method": "POST"
      }
    ]
  }
```

Get all email campaigns

This API is for getting a list of EMD email campaigns.

Service URL:

```
/rest/api/v1/campaigns
```

Request Method:

```
GET
```

Required Attributes:

```
None
```

```
Request Parameters:
      offset: starts at 0 and indicates the record number for the response result set
      limit: number of campaigns to return in the response (defaults to 200 and
     cannot exceed 200)
```

Request JSON Body:

```
None
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Response:

```
{
    "campaigns": [
        {
            "id": 1000,
            "name": "testcampaign-b11",
            "folderName": "testfolder",
            "type": "EMAIL",
```

```
"description": "<description>",
"purpose": "PROMOTIONAL",
"marketingStrategy": "<strategy>",
"marketingProgram": "<program>",
"listName": "<listname>",
"filterPaths": [
    "foldername/objectName1",
    "foldername/objectName2"
],
"refiningDataSourcePath": "foldername/objectName1",
"proofListPath": "foldername/objectName1",
"seedListPath": "foldername/objectName1",
"segmentPaths": [
    "foldername/objectName1",
    "foldername/objectName2"
],
"supplementaryCampaignDataSourcePaths": [
    "foldername/objectName1",
    "foldername/objectName2"
],
"supplementaryProofDataSourcePaths": [
    "foldername/objectName1",
    "foldername/objectName2"
],
"supplementarySeedDataSourcePaths": [
    "foldername/objectName1",
    "foldername/objectName2"
],
"suppressionListPaths": [
    "foldername/objectName1",
    "foldername/objectName2"
],
"subject": "<subject>",
"fromName": "<from name>",
"fromEmail": "<from email>",
"replyToEmail": "<reply to email>",
"bccEmail": "<bcc email>",
"htmlMessagePath": "documentPath",
"textMessagePath": "documentPath",
"enableExternalTracking": true,
"externalTrackingParams": {
    "name1": "value1",
    "name2": "value2"
},
"enableLinkTracking": true,
"linkTablePath": "foldername/objectName1",
"attachmentPaths": [
    "documentPath1",
    "documentPath2"
],
"campaignVariables": {
    "name1": "value1",
    "name2": "value2"
},
"useUTF8": true,
"locale": "<value>",
"trackHTMLOpens": true,
"trackConversions": true,
"sendTextIfHTMLUnknown": true,
"segmentTrackingColumnName": "<name>",
"unsubscribeOption": "OPTOUT_SINGLE_CLICK",
"unsubscribeFormName": "name",
"autoCloseOption": "NO_AUTO_CLOSE",
```

```
        "autoCloseValue": "<value>",
        "closedCampaignURL": "<URL>",
        "externalCampaignCode": "<code>",
        "salesForceCampaignId": "<salesforce id>",
        "links": [
            {
                "rel": "self",
                "href": "/rest/api/v1/campaigns/<campaign_name>",
                "method": "GET"
            },
            {
                "rel": "create",
                "href": "/rest/api/v1/campaigns",
                "method": "POST"
            },
            {
                "rel": "updateCampaign",
                "href": "rest/api/v1/campaigns/<campaign_name>",
                "method": "PUT"
            }
        ]
    },
    {
        "id": 1001,
        "name": "testcampaign-b12",
        "folderName": "testfolder",
        "type": "EMAIL",
        "description": "<description>",
        "purpose": "PROMOTIONAL",
        "marketingStrategy": "<strategy>",
        "marketingProgram": "<program>",
        "listName": "<listname>",
        "filterPaths": [
            "foldername/objectName1",
            "foldername/objectName2"
        ],
        "refiningDataSourcePath": "foldername/objectName1",
        "proofListPath": "foldername/objectName1",
        "seedListPath": "foldername/objectName1",
        "segmentPaths": [
            "foldername/objectName1",
            "foldername/objectName2"
        ],
        "supplementaryCampaignDataSourcePaths": [
            "foldername/objectName1",
            "foldername/objectName2"
        ],
        "supplementaryProofDataSourcePaths": [
            "foldername/objectName1",
            "foldername/objectName2"
        ],
        "supplementarySeedDataSourcePaths": [
            "foldername/objectName1",
            "foldername/objectName2"
        ],
        "suppressionListPaths": [
            "foldername/objectName1",
            "foldername/objectName2"
        ],
        "subject": "<subject>",
        "fromName": "<from name>",
        "fromEmail": "<from email>",
        "replyToEmail": "<reply to email>",
```

```
            "bccEmail": "<bcc email>",
            "htmlMessagePath": "documentPath",
            "textMessagePath": "documentPath",
            "enableExternalTracking": true,
            "externalTrackingParams": {
                "name1": "value1",
                "name2": "value2"
            },
            "enableLinkTracking": true,
            "linkTablePath": "foldername/objectName1",
            "attachmentPaths": [
                "documentPath1",
                "documentPath2"
            ],
            "campaignVariables": {
                "name1": "value1",
                "name2": "value2"
            },
            "useUTF8": true,
            "locale": "<value>",
            "trackHTMLOpens": true,
            "trackConversions": true,
            "sendTextIfHTMLUnknown": true,
            "segmentTrackingColumnName": "<name>",
            "unsubscribeOption": "OPTOUT_SINGLE_CLICK",
            "unsubscribeFormName": "name",
            "autoCloseOption": "NO_AUTO_CLOSE",
            "autoCloseValue": "<value>",
            "closedCampaignURL": "<URL>",
            "externalCampaignCode": "<code>",
            "salesForceCampaignId": "<salesforce id>",
            "links": [
                {
                    "rel": "self",
                    "href": "/rest/api/v1/campaigns/<campaign_name>",
                    "method": "GET"
                },
                {
                    "rel": "create",
                    "href": "/rest/api/v1/campaigns",
                    "method": "POST"
                },
                {
                    "rel": "updateCampaign",
                    "href": "rest/api/v1/campaigns/<campaign_name>",
                    "method": "PUT"
                }
            ]
        }
    ],
    "links": [
        {
            "rel": "self",
            "href": "/rest/api/v1/campaigns",
            "method": "GET"
        },
        {
            "rel": "prev",
            "href": "/rest/api/v1/campaigns?limit=200@offset=0",
            "method": "GET"
        },
        {
            "rel": "next",
```

```
            "href": "rest/api/v1/campaigns?limit=200@offset=200",
            "method": "GET"
        }
    ]
}
```

Managing Content Library Folders

The following interfaces are available to create a content library folder, delete a content library folder and list the contents of a content library folder.

Create content library folder

This interface is used to create a content library folder.

Service URL:

```
/rest/api/v1/clFolders
```

Request Method:

```
POST
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request JSON Body:

```
{
   "folderPath" : "<folderPath>"
}
```

Sample Response in case of success:

```
{
   "folderPath": "/contentlibrary/abn/f1/f2/f3",
   "links":    [
      {
         "rel": "self",
         "href": "/rest/api/v1/clFolders",
         "method": "POST"
      },
      {
         "rel": "listContentLibraryFolders",
         "href": "/rest/api/v1/clFolders/contentlibrary/abn/f1/f2/f3",
         "method": "GET"
      },
      {
         "rel": "deleteContentLibraryFolder",
         "href": "/rest/api/v1/clFolders/contentlibrary/abn/f1/f2/f3",
         "method": "DELETE"
      }
   ]
}
```

Sample Response in case of failure:

```
{
   "type": "",
   "title": "Folder already exists",
   "errorCode": "FOLDER_ALREADY_EXISTS",
   "detail": "/contentlibrary/abn/f1/f2/f3",
   "errorDetails": []
}
```

Delete content library folder

This interface is used to delete a content library folder.

Service URL:

```
/rest/api/v1/clFolders/<folderPath>
```

Request Method:

```
DELETE
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Sample Response in case of success:

```
{
   "folderPath": "/contentlibrary/abn/f1/f2/f3",
   "links":    [
      {
         "rel": "self",
         "href": "/rest/api/v1/clFolders/contentlibrary/abn/f1/f2/f3",
         "method": "DELETE"
      },
      {
         "rel": "createContentLibraryFolder",
         "href": "/rest/api/v1/clFolders",
         "method": "POST"
      }
   ]
}
```

Sample Response in case of failure:

```
{
   "type": "",
   "title": "Folder not found",
   "errorCode": "FOLDER_NOT_FOUND",
   "detail": "/contentlibrary/abn/f1/f2/f3",
   "errorDetails": []
}
```

List contents of a content library folder

This interface is used to list the contents of a content library folder. To get the objects at the root level, i.e., for the entire account, replace the folderPath parameter with the value ‘contentlibrary’.

Service URL:

```
/rest/api/v1/clFolders/<folderPath>?type=<objecttype>
```

Request Method:

```
GET
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request parameters:

```
type - Determines what content of a folder needs to be listed. Allowed values
       are 'all', 'folders', 'docs' or 'items'. Value defaults to 'all', so all
       contents of a folder need to be listed.
```

Sample Response in case of success:


```
{
   "folders":    [
          {
         "folderPath": "/contentlibrary/f1/sublevel-1",
         "links":          [
            {
               "rel": "listContentLibraryFolders",
               "href": "/rest/api/v1/clFolders/contentlibrary/f1/sublevel-1",
               "method": "GET"
            },
            {
               "rel": "deleteContentLibraryFolder",
               "href": "/rest/api/v1/clFolders/contentlibrary/f1/sublevel-1",
               "method": "DELETE"
            },
            {
               "rel": "createContentLibraryFolder",
               "href": "/rest/api/v1/clFolders",
               "method": "POST"
            }
         ]
      }wsrest_clwsrest_clwsrest_clwsrest_clwsrest_cl
   ],
   "documents":    [
          {
         "documentPath": "/contentlibrary/f1/doc2.htm",
         "content": null,
         "links":          [
            {
               "rel": "deleteDocument",
               "href": "/rest/api/v1/clDocs/contentlibrary/f1/doc2.htm",
               "method": "DELETE"
            },
            {
               "rel": "createDocument",
               "href": "/rest/api/v1/clDocs",
               "method": "POST"
            },
            {
               "rel": "setDocumentContent",
               "href": "/rest/api/v1/clDocs/contentlibrary/f1/doc2.htm",
               "method": "POST"
```

```
            },
            {
               "rel": "getDocumentContent",
               "href": "/rest/api/v1/clDocs/contentlibrary/f1/doc2.htm",
               "method": "GET"
            }
         ]
      }wsrest_clwsrest_clwsrest_clwsrest_clwsrest_clwsrest_clwsrest_clwsrest_cl
   ],
   "items":    [
            {
         "itemPath": "/contentlibrary/f1/testcreate2050.png",
         "itemData": null,
         "links":          [
            {
               "rel": "getContentLibraryItem",
               "href":
"/rest/api/v1/clItems/contentlibrary/f1/testcreate2050.png",
               "method": "GET"
            },
            {
               "rel": "setContentLibraryItem",
               "href":
"/rest/api/v1/clItems/contentlibrary/f1/testcreate2050.png",
               "method": "POST"
            },
            {
               "rel": "deleteContentLibraryItem",
               "href":
"/rest/api/v1/clItems/contentlibrary/f1/testcreate2050.png",
               "method": "DELETE"
            },
            {
               "rel": "createContentLibraryItem",
               "href": "/rest/api/v1/clItems",
               "method": "POST"
            }
         ]
      }
   ],
   "links":    [
            {
         "rel": "self",
         "href": "/rest/api/v1/clFolders/contentlibrary/f1",
         "method": "GET"
      },
      {
         "rel": "deleteContentLibraryFolder",
         "href": "/rest/api/v1/clFolders/contentlibrary/f1",
         "method": "DELETE"
      }
   ]
}
```


Sample Response in case of failure:


```
{
   "type": "",
   "title": "Invalid request parameters",
   "errorCode": "INVALID_PARAMETER",
   "detail": "Type should be either folders, items or docs",
   "errorDetails": []
```

```
}
```

Managing Content Library Documents

The following interfaces are available to create a content library document, update a content library document, retrieve the contents of a content library document and delete a content library document.

Create content library document

This interface is used to create a content library document. Please note that ONLY UTF-8 character encoding will be supported by the REST API.

Service URL:

```
/rest/api/v1/clDocs
```

Request Method:

POST

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request JSON Body:

```
{
    "documentPath": "/contentlibrary/abn/wsrest_cl.htm",
    "content": "<html dir=\"ltr\">\r\n <head>\r\n  <title><\/title>\r\n
 <\/head>\r\n <body>\r\n  <p>test document<\/p>\r\n  <p><img
 src=\"wsrest_cl.images/testcreate-1.png\" alt=\"\" /><\/p>\r\n
 <\/body>\r\n<\/html>\n\n"
}
```

NOTE: Special characters in the document content need to be UTF-8 encoded. If that is not the case, an error response is given.

Sample Response in case of success:

NOTE: Content Library saves files with ".html" extension with a ".htm" extension. Therefore the response will have links to the document with a ".htm" extension.

```
{
   "documentPath": "/contentlibrary/abn/wsrest_cl.htm",
   "content": null,
   "links":    [
            {
         "rel": "self",
         "href": "/rest/api/v1/clDocs",
         "method": "POST"
      },
            {
         "rel": "getDocumentContent",
         "href": "/rest/api/v1/clDocs/contentlibrary/abn/wsrest_cl.htm",
         "method": "GET"
      },
```

```
            {
            "rel": "deleteDocument",
            "href": "/rest/api/v1/clDocs/contentlibrary/abn/wsrest_cl.htm",
            "method": "DELETE"
        },
            {
            "rel": "setDocumentContent",
            "href": "/rest/api/v1/clDocs/contentlibrary/abn/wsrest_cl.htm",
            "method": "POST"
        }
    ]
}
```

Sample Response in case of failure:

```
{
    "type": "",
    "title": "Document already exists",
    "errorCode": "DOCUMENT_ALREADY_EXISTS",
    "detail": "/contentlibrary/abn/wsrest_cl.htm",
    "errorDetails": []
}
```

Retrieve contents of a content library document

This interface is used to retrieve the contents of a content library document.

Service URL:

```
/rest/api/v1/clDocs/<documentPath>
```

Request Method:

```
GET
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Sample Response in case of success:

NOTE: Response contains the content of the document as it is saved in the content library. WSAPI does not decode the content of the document before returning the response.

```
{
    "documentPath": "/contentlibrary/abn/wsrest_cl.htm",
    "content": "<html dir=\"ltr\">\r\n <head>\r\n  <title><\/title>\r\n
<\/head>\r\n <body>\r\n  <p>test document<\/p>\r\n  <p><img
src=\"wsrest_cl.images/testcreate-1.png\" alt=\"\" /><\/p>\r\n
<\/body>\r\n<\/html>\n\n\n",
    "links":    [
        {
            "rel": "self",
            "href": "/rest/api/v1/clDocs/contentlibrary/abn/wsrest_cl.htm",
            "method": "GET"
        },
        {
            "rel": "deleteDocument",
            "href": "/rest/api/v1/clDocs/contentlibrary/abn/wsrest_cl.htm",
```

```
            "method": "DELETE"
         },
         {
            "rel": "setDocumentContent",
            "href": "/rest/api/v1/clDocs/contentlibrary/abn/wsrest_cl.htm",
            "method": "POST"
         },
         {
            "rel": "createDocument",
            "href": "/rest/api/v1/clDocs",
            "method": "POST"
         }
      ]
   }
```

Sample Response in case of failure:

```
   {
      "type": "",
      "title": "Document not found",
      "errorCode": "DOCUMENT_NOT_FOUND",
      "detail": "/contentlibrary/abn/wsrest_cl.htm",
      "errorDetails": []
   }
```

Update contents of a content library document

This interface is used to update the contents of a content library document.

Service URL:

```
/rest/api/v1/clDocs/<documentPath>
```

Request Method:

```
POST
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request JSON Body:

```
{
   "documentPath" : "/contentlibrary/abn/wsrest_cl.htm",
   "content": "test documentersasefgwdfgsdfg"
}
```

Sample Response in case of success:

Note: The "content" attribute in the response is returned as null always. This is done to avoid returning large contents in the response.

```
   {
      "documentPath": "/contentlibrary/abn/wsrest_cl.htm",
```

```
   "content": null,
   "links":    [
      {
         "rel": "self",
         "href": "/rest/api/v1/clDocs/contentlibrary/abn/wsrest_cl.htm",
         "method": "POST"
      },
      {
         "rel": "getDocumentContent",
         "href": "/rest/api/v1/clDocs/contentlibrary/abn/wsrest_cl.htm",
         "method": "GET"
      },
      {
         "rel": "deleteDocument",
         "href": "/rest/api/v1/clDocs/contentlibrary/abn/wsrest_cl.htm",
         "method": "DELETE"
      },
      {
         "rel": "createDocument",
         "href": "/rest/api/v1/clDocs",
         "method": "POST"
      }
   ]
}
```

Sample Response in case of failure:

```
{
   "type": "",
   "title": "Invalid request parameters",
   "errorCode": "INVALID_PARAMETER",
   "detail": "Document Path in the URI does not match Document Path in the
request payload",
   "errorDetails": []
}
```

Delete a content library document

This interface is used to delete a content library document.

Service URL:

```
/rest/api/v1/clDocs/<documentPath>
```

Request Method:

```
DELETE
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Sample Response in case of success:

```
{
   "documentPath": "/contentlibrary/abn/wsrest_cl.htm",
   "content": null,
   "links":    [
     {
```

```
            "rel": "self",
            "href": "/rest/api/v1/clDocs/contentlibrary/abn/wsrest_cl.htm",
            "method": "DELETE"
         },
         {
            "rel": "createDocument",
            "href": "/rest/api/v1/clDocs",
            "method": "POST"
         }
      ]
   }
```

Sample Response in case of failure:

```
{
   "type": "",
   "title": "Document not found",
   "errorCode": "DOCUMENT_NOT_FOUND",
   "detail": "/contentlibrary/abn/wsrest_cl.htm",
   "errorDetails": []
}
```

Create a copy of a content library document

This interface is used to create a copy of a content library document.

Service URL:

```
/rest/api/v1/clDocs/<destinationDocumentPath>
```

Request Method:

```
PUT
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request JSON Body:

```
{
     "documentPath": "<sourceDocumentPath>"
}
```

Sample Response in case of success:

Note: The "content" attribute in the response is returned as null always. This is done to avoid returning large contents in the response.

```
{
   "documentPath": "/contentlibrary/abn/doc22.htm",
   "content": null,
   "links":    [
```

```
        {
            "rel": "self",
            "href": "/rest/api/v1/clDocs/contentlibrary/abn/doc22.htm",
            "method": "PUT"
         },
         {
            "rel": "getDocumentContent",
            "href": "/rest/api/v1/clDocs/contentlibrary/abn/doc22.htm",
            "method": "GET"
         },
         {
            "rel": "deleteDocument",
            "href": "/rest/api/v1/clDocs/contentlibrary/abn/doc22.htm",
            "method": "DELETE"
         }
      ]
}
```

Sample Response in case of failure:

```
{
   "type": "",
   "title": "Document not found",
   "errorCode": "DOCUMENT_NOT_FOUND",
   "detail": "Document not found:/contentlibrary/abn/wsrest_cl.htm",
   "errorDetails": []
}
```

Managing Content Library Media Files

The following interfaces are available to create a content library media file in a folder, update a content library media file, retrieve the contents of a content library media file and delete a content library media file.

Create content library media file

This interface is used to create a content library media file in a content library folder.

Service URL:

```
/rest/api/v1/clItems
```

Request Method:

```
POST
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request JSON Body:

```
{
  "itemPath": "/contentlibrary/abn/testcreate_50.png",
```

```
  "itemData": "<base64 encoded binary string>"
}
```

Sample Response in case of success:

Note: The "itemData" attribute in the response is returned as null always. This is done to avoid returning large binary content in the response.

```
{
   "itemPath": "/contentlibrary/abn/testcreate_50.png",
   "itemData": null,
   "links":    [
      {
         "rel": "self",
         "href": "/rest/api/v1/clItems",
         "method": "POST"
      },
      {
         "rel": "getContentLibraryItem",
         "href": "/rest/api/v1/clItems/contentlibrary/abn/testcreate_50.png",
         "method": "GET"
      },
      {
         "rel": "deleteContentLibraryItem",
         "href": "/rest/api/v1/clItems/contentlibrary/abn/testcreate_50.png",
         "method": "DELETE"
      },
      {
         "rel": "setContentLibraryItem",
         "href": "/rest/api/v1/clItems/contentlibrary/abn/testcreate_50.png",
         "method": "POST"
      }
   ]
}
```

Sample Response in case of failure:

```
{
   "type": "",
   "title": "Document already exists",
   "errorCode": "DOCUMENT_ALREADY_EXISTS",
   "detail": "/contentlibrary/abn/testcreate_50.png",
   "errorDetails": []
}
```

Retrieve contents of a content library media file

This interface is used to retrieve the contents of a content library media file.

Service URL:

```
/rest/api/v1/clItems/<itemPath>
```

Request Method:

```
GET
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Sample Response in case of success:

Note: "itemData" attribute in the response is a BASE64 encoded binary string representation of the media file content.


```
{
   "itemPath": "/contentlibrary/abn/testcreate_50.png",
   "itemData":
"iVBORw0KGgoAAAANSUhEUgAAAdwAAAFUCAIAAAABIhoXAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8
YQUAAAAJcEhZcwAAFiUAABYlAUlSJPAAAA6XSURBVHhe7d0vVBtLG8BhZCQyEhmJRCKRkUgkEhkXiYy8M
hKJREYiI5GRkZHIfnO60zn7hT9N20z2TfZ5xD3NLoWEc/tjmMzOnv0AIAxRBghElAECEWWAQEQZIBBRBg
hElAECEWWAQEQZIBBRBghElAECEWWAQEQZIBBRBghElAECEWWAQEQZIBBRBghElAECEWWAQEQZIBBRBghE
lAECEWWAQEQZIBBRBghElAECEWWAQEQZIBBRBghElAECEWWAQEQZIBBRBghElAECEWWAQEQZIBBRBghEl
AECEWWAQEQZIBBRBghElAECEWWAQEQZIBBRBghElAECEWWAQEQZIBBRBghElAECEWWAQEQZIBBRBghElA
ECEWWAQEQZIBBR7sDLy8v19fV8Ps+PAX4R5Q4Mh8Ozs7PBYJAfA/wiyh1IOU5RTqbTaT4E8JMod+Dh4aG
JcqLLQJsod+D9/f3m5iZX+exssVjkE/uz2WweHx8vLi7y1ziU9BXv7+9rvCLoCVHuRury9fV1CVl6mE/8
s9VqlbJYZkg6l17df//9l58c8Dui3Jn1en1+ft6UK2U0H/0HaXw6Ho+bTxiNNMOORLlL8/k8R+vsbLlc5
qN/Ln2ey8vL/Il+SSPx5+fn/BGHkl7Fw8PDV9MmJtDht0S5Y2VyOQ1y86GdNQVsFti13d3d/Uvi9+j9/T
09mfy0fjFqhm+IcsdSPXOrdh4sfzUaPT8/n0wm6/U6f1wYW29sFuklpBcS8AlDh0S5e2Ui+PvB8lctTpq
x5x7fLdy7T4fMbYbP0BDl7v12sPzV4rbhcHh068++r7NJZxDlEL4aLL+9vV1dXTWnimNs8aeen58/vj+Z
fgLl09BLohxCe7BcNipKeWovNz6ZFn/UnnS2JQg9J8pR3N7eliqlIWR7gJyOnPz4MXU5v9oz/0/Sa/4BR
LHZbD6dOE51fnt7yx900vILFmX6zT+AQF5fX9vzFX0YIBfT6TS/bFGm3/wDCKQdpqSfRb69vc1HoZdEOY
qtIidppNyHiYvJZJJf8NnZzc1N5NXWcACiHEK7yClMZXJ5NBqdcKS2FvwpMiSi3L2tIqcwtSeX7+7u8se
dlvl83p5AV2RoiHLHPha5OT6bzfLRs7PTu/64vT1er97PhN8S5S6l2uYyfTZULJf5pWylsXM+evzaRe7P
gj/YkSh3qcwdf/rLezpSrkIeDoensZtau8imLOAjUe5MimzTpvPz86/atFqtyt1J0qDy2BOmyPBbotyZ5
+fnJk/X19f50GcWi0X7Tb/jDZkiwy5EuTPlLb6Hh4d86AvtN/1Go9Exzi8rMuxIlDtT3sd7enrKh752f3
/ffHBjMpnkE8eg/UNFkeF7otyZMlm84/KDNNgsfyW5vr7ebDb5XGDp94D8jBUZdiDK3SgbKKfO5kM7WK/
XqWvNX0wuLi4+vVNJEKm/ZT/SJP1moMjwW6LcjfIb/V9csNe+3iQ1/fn5OZ+IJI3i01g+P8uzs/v7+3wC
+JYod6MMeOe/7jPyR1KI21MZk8kk1CB0tVq17/Pkgj3YnSh3IAW0rHJL/cpH/9Db29toNGo+SXJxcRFky
LxcLstFMell/t1PHegtUe7AYrFompXilQ/9lc1mU5ZwNNIAvNurltNLK0P49IeXl5d8AtiNKHegTArvZa
b16elpOBw2n7DRyWxG+oqPrTu9pqcU+U1ICEuUO1DmW/c14ZCGzO2t4pMDX2Myn8/bPxjSV//raRnoOVE
+tBTQnK6zs/0uNH57e2sveEgOMGReLBbt9/SSq6ur/b4u6BVRPrSy5UVqWT60V1vXmNQbMqefAe1F00ka
LJ/e1s9wYKJ8aHd3d03C6l0qvXWNSXJ7e5tyua/3ANPn37rsezAYTKfTUMvy4EiJ8qGV5WLpF/98qI6tI
XM9KdCnsdczRCDKB7VarZqQpaHlAcaVH4fM+5U+uSUWsF+ifFDl/k/f76G8X6mbs9lsPB7vceB8eXlpDT
LUIMoHVa71mE6n+RBAiygfzvv7exmr+q0f+JQoH05ZDDccDvMhgP8nyodTNheutxgOOHaifCCbzabsC2H
uAviKKB/I/NedQ0ejUT4E8IEoH0jZlcKO78A3RPkQ1ut1U+TE9mnAN0T5ENLouCnyIa8ZAY6RKB9C2dzS
JmrA90S5ure3t6bIg8HARsPA90S5unJPkPF4nA8BfEGUqyt7dQa52zQQmSjXVW5cfX5+bg944LdEua5yh
4693LgaOHmiXNfB7jMCnAZRrqvceD8/BviWWNTVFDnJjwG+JRZ1NUW+uLjIjwG+JcoVldukivLfmc/n19
fX6b/5MfSAKFdUonx5eZkPsbPpdNp89w5z528IQpQrWi6XTVbsQ/Snym2/k9vb23wUekCUKypXjtzc3OR
D7GC9Xpe7tKRvnWEyvSLKFb28vDRlubu7y4fYwcPDQ/N9u7y8VGT6RpQrKreAEuXdtYfJdguhh0S5ohJl
t6/eXfoB1nzTvDtKP4lyReWGI9PpNB/iay8vL6PRqPmOJYbJ9JMoV1QWdc1ms3yIzyyXy3Jj2Ya3RuktU
a6ovGHl8oevrNfrspFeYzgcumkWfSbKFZXpUVH+1OPj4/n5efMtSgaDwWQycccsek6UKypRtm/nlre3t6
urq+ab0xiPx6vVKp+GHhPliso8qSi3pQFyWfSWXF5e+v5AIcoVlSingWE+1G/r9bo9QE5pToHO54CfRLm
issDLL+bJfD5vzyCnOvtZBR+JckXlXlBphJgP9VJ6+Tc3N823IjFAhm+IckWlQflxL81mMwNk2J0o11I2
Ux6NRvlQz2xdoWeADLsQ5Vr6vG9nGgu35ysSA2TYkSjXUnYjur+/z4d6YLPZlOsYG67Qgz8iyrWUjS/68
zv71vTxwBV68OdEuZZyOd/T01M+dLq2po8TV+jB3xHlWsqVI6+vr/nQKfo4fewKPfgXolzLyS9SNn0MNY
hyLU2nTnWR8tPTk+ljqEGUqzjtRcqpv82ra5g+hj0S5SpOdZHy+/v77e1t89KS9CPH9DHslyhXcZKLlDe
bTXuPtzRANl8BeyfKVZzeIuXVatVe9Pbw8JBPAHslylWc2CLl19fX9tt6llhAPaJcxSktUk4/Vwa/bhSS
0vzy8pJPABWIchUns0h5Nps1LyRJL8qmQlCbKFdRopwfH6f2tSFXV1fe1oMDEOUqjj3KW0vfxuNxOpLPA
TWJchVHHeU0Ii5z4kmvth6FzolyFccb5dVqdXl52Tz5xL1C4MBEuYojjfJyuSzPfDAYzOfzfAI4FFGu4h
ijvFgsymJkS9+gK6JcxdFFeTablcXIw+EwDZnzCeCwRLmKI4ry1k1D0p9t+QYdEuUqjiLKn95z2mJk6JY
oVxE8yqm8bhoCMYlyFZGjvHXP6cRNQyAOUa4iZpQ/3nP65ubGdhYQiihXUYai+XHXlsvl1vRxqrNFbxCQ
KFeRyxcgyuv1+v7+Pj+bn9IPjNlslk8DwYhyFbl/XUf54/Txw8OD6WOITJSryAnsLsqpvFvzFaaP4SiIc
hU5hB1FebFYDIfD/AxMH8NREeUqcg67iPJkMslf+6f7+3tbIcMREeUqchEPG+XVanV1dZW/8M839J6fn/
M54EiIchW5iweM8tPTU/s9vVTnY789IPSTKFeR03iQKL+/v9/d3eWv99N0Os3ngGMjylXkOtaP8nK5bF+
kNxwOX19f8zngCIlyFbmRlaM8n8/LJsjJeDy2BhmOnShXUVpZb+u1l5eX5ksk6cvZ4w1OgyhXUTbGTLms
```

```
MZ+wXC5L90ejkatC4GSIchXv7+9lddpwONzvQojValWuDbm4uLDKAk6JKNfSTmcK9L6u4NhsNuWdvfPzc
2NkODGiXNFisSiTDHd3d//e5fYAPH3m9PnzCeBUiHJds9msaWiSRrj/OL88Ho/z5zo7c7UenCRRrm5rO+
PHx8d84g+176pnQ2Q4VaJ8CPP5vH0NdHqYT+xsOp3mv/xzT+R8FDg5onwg6/X6+vq6qepgMPijN+jaRR6
Px/kocIpE+XA2m025oepoNNrxfb92kW9ubva1igOISZQP6vX1tb0eIx/9miJD34jyobXXY3x/QxBFhh4S
5Q6UlW3D4fCrLYQUGfpJlDuwXq/b10lPJpOt5ioy9JYod+Pp6SlH96fmupLVajWfz29vb/NRRYb+EeXOb
N3h9CNFhh4S5Y5tXVdSKDL0kyh3b71epwSnEA8Gg/SHx8dHt3SC3hJlgEBEGSAQUQYIRJQBAhFlgEBEGS
AQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSA
QUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQU
QYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQY
IRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIR
JQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgDB+/PgfhFa/3C
47xMoAAAAASUVORK5CYII=",
   "links":    [
            {
         "rel": "self",
         "href": "/rest/api/v1/clItems/contentlibrary/abn/testcreate_50.png",
         "method": "GET"
      },
            {
         "rel": "deleteContentLibraryItem",
         "href": "/rest/api/v1/clItems/contentlibrary/abn/testcreate_50.png",
         "method": "DELETE"
      },
            {
         "rel": "setContentLibraryItem",
         "href": "/rest/api/v1/clItems/contentlibrary/abn/testcreate_50.png",
         "method": "POST"
      },
            {
         "rel": "createContentLibraryItem",
         "href": "/rest/api/v1/clItems",
         "method": "POST"
      }
   ]
}
```


Sample Response in case of failure:

```
{
   "type": "",
   "title": "Document not found",
   "errorCode": "DOCUMENT_NOT_FOUND",
   "detail": "/contentlibrary/abn/testcreate.png",
   "errorDetails": []
}
```

Update contents of a content library media file

This interface is used to update the contents of a content library media file.

Service URL:

```
/rest/api/v1/clItems/<itemPath>
```

Request Method:

```
POST
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request JSON Body:

NOTE: Since the payload accepts binary content, the entire content of the existing media file will be replaced with the content provided in the request payload.

```
{
  "itemPath" : "/contentlibrary/abn/testcreate.png",
  "itemData": "<base64 encoded binary string>”
}
```

Sample Response in case of success:

Note: The “itemData” attribute in the response is returned as null always. This is done to avoid returning large binary content in the response.

```
{
   "itemPath": "/contentlibrary/abn/testcreate_50.png",
   "itemData": null,
   "links":    [
            {
         "rel": "self",
         "href": "/rest/api/v1/clItems/contentlibrary/abn/testcreate_50.png",
         "method": "POST"
      },
            {
         "rel": "getContentLibraryItem",
         "href": "/rest/api/v1/clItems/contentlibrary/abn/testcreate_50.png",
         "method": "GET"
      },
            {
         "rel": "deleteContentLibraryItem",
         "href": "/rest/api/v1/clItems/contentlibrary/abn/testcreate_50.png",
         "method": "DELETE"
      },
            {
         "rel": "createContentLibraryItem",
         "href": "/rest/api/v1/clItems",
         "method": "POST"
      }
   ]
}
```

Sample Response in case of failure:

```
{
   "type": "",
   "title": "Invalid request parameters",
   "errorCode": "INVALID_PARAMETER",
   "detail": "Item Path in the URI does not match Item Path in the request
payload",
   "errorDetails": []
```

```
}
```

Delete a content library media file

This interface is used to delete a content library media.

Service URL:

```
/rest/api/v1/clItems/<itemPath>
```

Request Method:

```
DELETE
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Sample Response in case of success:

```
{
   "itemPath": "/contentlibrary/abn/testcreate_50.png",
   "itemData": null,
   "links":    [
            {
         "rel": "self",
         "href": "/rest/api/v1/clItems/contentlibrary/abn/testcreate_50.png",
         "method": "DELETE"
      },
            {
         "rel": "createContentLibraryItem",
         "href": "/rest/api/v1/clItems",
         "method": "POST"
      }
   ]
}
```

Sample Response in case of failure:

```
{
   "type": "",
   "title": "Document not found",
   "errorCode": "DOCUMENT_NOT_FOUND",
   "detail": "Document not found:/contentlibrary/abn/testcreate_50.png",
   "errorDetails": []
}
```

Create a copy of a content library media file

This interface is used to create a copy of a content library media file.

Service URL:

```
/rest/api/v1/clItems/<destinationItemPath>
```

Request Method:

PUT

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request JSON Body:

```
{
    "itemPath" : "<sourceItemPath>"
}
```

Sample Response in case of success:

Note: The "itemData" attribute in the response is returned as null always. This is done to avoid returning large binary content in the response.


```
{
   "itemPath": "/contentlibrary/abn/copiedimage.png",
   "itemData": null,
   "links":    [
            {
         "rel": "self",
         "href": "/rest/api/v1/clItems/contentlibrary/abn/copiedimage.png",
         "method": "PUT"
      },
            {
         "rel": "getContentLibraryItem",
         "href": "/rest/api/v1/clItems/contentlibrary/abn/copiedimage.png",
         "method": "GET"
      },
            {
         "rel": "deleteContentLibraryItem",
         "href": "/rest/api/v1/clItems/contentlibrary/abn/copiedimage.png",
         "method": "DELETE"
      }
   ]
}
```


Sample Response in case of failure:


```
{
   "type": "",
   "title": "Document not found",
   "errorCode": "DOCUMENT_NOT_FOUND",
   "detail": "Document not found:/contentlibrary/abn/testcreate_50.png",
   "errorDetails": []
}
```

Managing Images of Content Library Documents

The following interfaces are to get images and set images in a content library document.

Set images in a content library document

This interface is used to set images in a content library document.

Service URL:

```
/rest/api/v1/clDocImages/<documentPath>
```

Request Method:

```
POST
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request JSON Body:

```
{
"documentPath" : "/contentlibrary/abn/wsrest_cl.htm",
"imageData": [
       {
         "itemPath": "testcreate.png",
         "itemData": "<base64 encoded binary string>”
       }
  ]
}
```

Sample Response in case of success:

```
{
   "documentPath": "/contentlibrary/abn/wsrest_cl.htm",
   "content": null,
   "links":    [
            {
         "rel": "self",
         "href": "/rest/api/v1/clDocImages/contentlibrary/abn/wsrest_cl.htm",
         "method": "POST"
      },
            {
         "rel": "getDocumentImages",
         "href": "/rest/api/v1/clDocImages/contentlibrary/abn/wsrest_cl.htm",
         "method": "GET"
      }
   ]
}
```

Sample Response in case of failure:

```
{
```

```
   "type": "",
   "title": "Invalid request parameters",
   "errorCode": "INVALID_PARAMETER",
   "detail": "Document Path in the URI does not match Document Path in the
request payload",
   "errorDetails": []
}
```

Get images in a content library document

This interface is used to get images in a content library document.

Service URL:

```
/rest/api/v1/clDocImages/<documentPath>
```

Request Method:

```
GET
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Sample Response in case of success:

```
{
   "documentPath": "/contentlibrary/abn/wsrest_cl.htm",
   "imageData": [   {
      "itemPath": "/contentlibrary/abn/testcreate.png",
      "itemData": "<base64 encoded binary string>",
      "links":       [
                  {
            "rel": "getContentLibraryItem",
            "href": "/rest/api/v1/clItems/contentlibrary/abn/testcreate.png",
            "method": "GET"
         },
                  {
            "rel": "setContentLibraryItem",
            "href": "/rest/api/v1/clItems/contentlibrary/abn/testcreate.png",
            "method": "POST"
         },
                  {
            "rel": "deleteContentLibraryItem",
            "href": "/rest/api/v1/clItems/contentlibrary/abn/testcreate.png",
            "method": "DELETE"
         },
                  {
            "rel": "createContentLibraryItem",
            "href": "/rest/api/v1/clItems",
            "method": "POST"
         }
      ]
   }],
   "links":    [
            {
         "rel": "self",
         "href": "/rest/api/v1/clDocImages/contentlibrary/abn/wsrest_cl.htm",
         "method": "GET"
```

```
        },
              {
           "rel": "setDocumentImages",
           "href": "/rest/api/v1/clDocImages/contentlibrary/abn/wsrest_cl.htm",
           "method": "POST"
        }
     ]
}
```

Sample Response in case of failure:

```
{
   "type": "",
   "title": "Images not found",
   "errorCode": "IMAGES_NOT_FOUND",
   "detail": "There are no images in wsrest_cl.htm",
   "errorDetails": []
}
```

Handling Errors

If a REST API request fails, the following error information is returned instead of the response described in previous sections. The format and description of the error codes are shown below.

Error returned:

```
{
 "type" : "<JSON_SCHEMA_LINK_FOR_THIS_ERROR_FOR_FUTURE>",
 "title" : "<ERROR_TITLE>",
 "errorCode" : "<ERROR_CODE>",
 "detail" : "<DETAIL_MESSAGE>",
 "errorDetails" : <FOR_FUTURE_USE>
}
```

Error codes:

ERROR_CODE	HTTP response code	Title
DUPLICATE_API_REQUEST	HttpStatus.BAD_REQUEST	Duplicate API request
API_DISABLED_FOR_USER	HttpStatus.UNAUTHORIZED	API disabled for user
INSUFFICIENT_ACCESS	HttpStatus.UNAUTHORIZED	Insufficient access
INVALID_USER_NAME_PASSWORD	HttpStatus.BAD_REQUEST	Invalid username or password
INVALID_SESSION_ID	HttpStatus.BAD_REQUEST	
INVALID_SOAP_HEADER	HttpStatus.BAD_REQUEST	
INVALID_NUMBER	HttpStatus.BAD_REQUEST	Invalid number
INVALID_DATE	HttpStatus.BAD_REQUEST	Invalid date
INVALID_PARAMETER	HttpStatus.BAD_REQUEST	Invalid Request JSON Body
INVALID_FIELD_NAME	HttpStatus.BAD_REQUEST	Invalid field name
INVALID_OBJECT	HttpStatus.BAD_REQUEST	Invalid object
PASSWORD_LOCKOUT	HttpStatus.UNAUTHORIZED	Password locked
PASSWORD_EXPIRED	HttpStatus.UNAUTHORIZED	Password expired
REQUEST_LIMIT_EXCEEDED	HttpStatus.BAD_REQUEST	Request limit exceeded
CAMPAIGN_NOT_FOUND	HttpStatus.NOT_FOUND	Campaign not found
CAMPAIGN_ALREADY_EXISTS	HttpStatus.BAD_REQUEST	Campaign already exists
RECIPIENT_LIMIT_EXCEEDED	HttpStatus.BAD_REQUEST	Recipient limit exceeded
MAX_ATTACHMENT_SIZE_EXCEEDED	HttpStatus.BAD_REQUEST	Attachment size exceeded

ERROR_CODE	HTTP response code	Title
CAMPAIGN_NOT_LISTENING	HttpStatus.BAD_REQUEST	Campaign not listening
CAMPAIGN_IS_INVALID	HttpStatus.BAD_REQUEST	Not a valid campaign
MOBILE_CAMPAIGN_DISABLED_FOR_USER	HttpStatus.BAD_REQUEST	Campaign disabled for user
FOLDER_NOT_FOUND	HttpStatus.NOT_FOUND	Folder not found
FOLDER_ALREADY_EXISTS	HttpStatus.BAD_REQUEST	Folder already exists
NO_CAMPAIGNS_IN_THIS_FOLDER	HttpStatus.BAD_REQUEST	No campaigns for this folder
NO_OBJECTS_IN_THIS_FOLDER	HttpStatus.BAD_REQUEST	No objects in this folder
LIST_NOT_FOUND	HttpStatus.NOT_FOUND	List not found
LIST_ALREADY_EXISTS	HttpStatus.BAD_REQUEST	List already exists
TABLE_NOT_FOUND	HttpStatus.NOT_FOUND	Table not found
CUSTOM_EVENT_NOT_FOUND	HttpStatus.NOT_FOUND	Custom event not found
RECORD_LIMIT_EXCEEDED	HttpStatus.BAD_REQUEST	Record limit exceeded
RECORD_NOT_FOUND	HttpStatus.NOT_FOUND	Record not found
OBJECT_NOT_FOUND	HttpStatus.NOT_FOUND	Object not found
OBJECT_ALREADY_EXISTS	HttpStatus.BAD_REQUEST	Object already exists
OPERATION_NOT_SUPPORTED	HttpStatus.FORBIDDEN	Operation not supported
MULTIPLE_OBJECTS_FOUND	HttpStatus.BAD_REQUEST	Multiple objects found
DOCUMENT_NOT_FOUND	HttpStatus.NOT_FOUND	Document not found
DOCUMENT_ALREADY_EXISTS	HttpStatus.BAD_REQUEST	Document already exists
IMAGES_NOT_FOUND	HttpStatus.NOT_FOUND	Images not found
UNEXPECTED_EXCEPTION	HttpStatus.INTERNAL_SERVER_ERROR	Unexpected exception
UNRECOVERABLE_EXCEPTION	HttpStatus.INTERNAL_SERVER_ERROR	Unrecoverable exception
INVALID_AUTHENTICATION_OPTION	HttpStatus.BAD_REQUEST	Invalid authentication option
AUTHENTICATION_FAILED	HttpStatus.UNAUTHORIZED	Authentication failed

ERROR_CODE	HTTP response code	Title
CLIENT_CERTIFICATE_EXPIRED	HttpStatus.UNAUTHORIZED	Client certificate expired
CLIENT_CERTIFICATE_NOT_YET_VALID	HttpStatus.UNAUTHORIZED	Client certificate not valid
CLIENT_CERTIFICATE_NOT_FOUND	HttpStatus.UNAUTHORIZED	Client certificate not found
SERVER_CERTIFICATE_EXPIRED	HttpStatus.UNAUTHORIZED	Server certificate expired
SERVER_CERTIFICATE_NOT_YET_VALID	HttpStatus.UNAUTHORIZED	Server certificate not valid yet
SERVER_CERTIFICATE_NOT_FOUND	HttpStatus.UNAUTHORIZED	Server certificate not found
SERVICE_UNAVAILABLE	HttpStatus.SERVICE_UNAVAILABLE	Service unavailable
TOKEN_EXPIRED	HttpStatus.UNAUTHORIZED	Authentication token expired
METHOD_NOT_SUPPORTED	HttpStatus.METHOD_NOT_ALLOWED	Method not supported
RESOURCE_NOT_FOUND	HttpStatus.NOT_FOUND	Resource not found
INVALID_REQUEST_CONTENT	HttpStatus.BAD_REQUEST	Invalid request content
INVALID_TOKEN	HttpStatus.UNAUTHORIZED	Not a valid token
PRIVATE_KEY_NOT_FOUND	HttpStatus.UNAUTHORIZED	Private key not found
NO_RECIPIENT_FOUND	HttpStatus.NOT_FOUND	No recipient found
MULTIPLE_RECIPIENTS_FOUND	HttpStatus.BAD_REQUEST	Multiple recipients found
RECIPIENT_STATUS_UNDELIVERABLE	HttpStatus.OK	Recipient status undeliverable
PROFILE_LIST_NOT_FOUND	HttpStatus.NOT_FOUND	Profile list not found
PROFILE_LIST_NOT_FOUND_IN_FOLDER	HttpStatus.NOT_FOUND	Profile list not found in folder
USER_BLOCKED	HttpStatus.UNAUTHORIZED	User is blocked
LOGINS_DISABLED	HttpStatus.UNAUTHORIZED	Login is disabled
INVALID_AUTH_OPTION	HttpStatus.UNAUTHORIZED	Not a valid authentication option
SERVER_CHALLENGES_DO_NOT_MATCH	HttpStatus.UNAUTHORIZED	Server challenge did not match
INACTIVE_ACCOUNT	HttpStatus.UNAUTHORIZED	Account is not active

ERROR_CODE	HTTP response code	Title
MAX_CONCURRENT_SESSIONS_EXCEEDED	HttpStatus.UNAUTHORIZED	Maximum concurrent sessions exceeded
MAX_LOGIN_FAILURES_EXCEEDED	HttpStatus.UNAUTHORIZED	Maximum login failure exceeded
LOGIN_BLOCKED_TEMPORARILY	HttpStatus.UNAUTHORIZED	Login is blocked temporarily
ACCOUNT_SUSPENDED	HttpStatus.UNAUTHORIZED	Account is in suspended state
LOGIN_BLOCKED_INVALID_IPRANGE	HttpStatus.UNAUTHORIZED	Not a valid client IP range
WS_ARG_DISABLED	HttpStatus.UNAUTHORIZED	Web service ARG disabled
CAMPAIGN_IS_INVALID	HttpStatus.BAD_REQUEST	Campaign name is invalid.
CAMPAIGN_NOT_FOUND	HttpStatus.NOT_FOUND	Campaign not found
UNABLE_TO_CREATE_CAMPAIGN	HttpStatus.INTERNAL_SERVER_ERROR	Unable to create campaign
MOBILE_CAMPAIGN_DISABLED_FOR_USER	HttpStatus.BAD_REQUEST	Campaign disabled for user
CAMPAIGN_ALREADY_EXISTS	HttpStatus.BAD_REQUEST	Campaign already exists
PATH_NOT_VALID	HttpStatus.BAD_REQUEST	Object path not valid.
CAMPAIGN_SCHEDULE_DUPLICATE	HttpStatus.BAD_REQUEST	Campaign is already scheduled for launch
CURRENT_CAMPAIGN_SCHEDULE_AT_SAME_TIME	HttpStatus.BAD_REQUEST	This Schedule is already scheduled to run at specified time
CAMPAIGN_LAUNCH_SCHEDULE_DATE_PAST	HttpStatus.BAD_REQUEST	Campaign launch date is past
INVALID_CAMPAIGN_SCHEDULE_TYPE_CHANGE	HttpStatus.BAD_REQUEST	Campaign Schedule Type cannot be changed

ERROR_CODE	HTTP response code	Title
CAMPAIGN_SCHEDULE_NOT_FOUND	HttpStatus.BAD_REQUEST	Campaign Schedule not found
CAMPAIGN_NAME_NOT_VALID	HttpStatus.BAD_REQUEST	Campaign Name is not valid
INVALID_CAMPAIGN_SCHEDULE_TYPE_CHANGE	HttpStatus.BAD_REQUEST	Campaign Schedule Type cannot be changed
INVALID_CAMPAIGN_SCHEDULE_TYPE	HttpStatus.BAD_REQUEST	Invalid Campaign Schedule Type
CAMPAIGN_LAUNCH_ALREADY_HAPPENED	HttpStatus.BAD_REQUEST	Campaign launch has already occurred.
INVALID_CAMPAIGN_SCHEDULE_TIME	HttpStatus.BAD_REQUEST	Invalid Schedule Time.
CAMPAIGN_IS_INVALID	HttpStatus.BAD_REQUEST	Campaign name is invalid.
CAMPAIGN_NOT_FOUND	HttpStatus.NOT_FOUND	Campaign not found
UNABLE_TO_CREATE_CAMPAIGN	HttpStatus.INTERNAL_SERVER_ERROR	Unable to create campaign
MOBILE_CAMPAIGN_DISABLED_FOR_USER	HttpStatus.BAD_REQUEST	Campaign disabled for user
CAMPAIGN_ALREADY_EXISTS	HttpStatus.BAD_REQUEST	Campaign already exists
PATH_NOT_VALID	HttpStatus.BAD_REQUEST	Object path not valid.
CAMPAIGN_SCHEDULE_DUPLICATE	HttpStatus.BAD_REQUEST	Campaign is already scheduled for launch
CURRENT_CAMPAIGN_SCHEDULE_AT_SAME_TIME	HttpStatus.BAD_REQUEST	This Schedule is already scheduled to run at specified time

ERROR_CODE	HTTP response code	Title
CAMPAIGN_LAUNCH_SCHEDULE_DATE_PAST	HttpStatus.BAD_REQUEST	Campaign launch date is past
INVALID_CAMPAIGN_SCHEDULE_TYPE_CHANGE	HttpStatus.BAD_REQUEST	Campaign Schedule Type cannot be changed
CAMPAIGN_SCHEDULE_NOT_FOUND	HttpStatus.BAD_REQUEST	Campaign Schedule not found
CAMPAIGN_NAME_NOT_VALID	HttpStatus.BAD_REQUEST	Campaign Name is not valid
INVALID_CAMPAIGN_SCHEDULE_TYPE_CHANGE	HttpStatus.BAD_REQUEST	Campaign Schedule Type cannot be changed
INVALID_CAMPAIGN_SCHEDULE_TYPE	HttpStatus.BAD_REQUEST	Invalid Campaign Schedule Type
CAMPAIGN_LAUNCH_ALREADY_HAPPENED	HttpStatus.BAD_REQUEST	Campaign launch has already occurred.
INVALID_CAMPAIGN_SCHEDULE_TIME	HttpStatus.BAD_REQUEST	Invalid Schedule Time.

Definitions of Rule Parameters for Merging Members into a Profile List

Name	Type	Description
insertOnNoMatch	boolean	Indicates what should be done for records where a match is not found (true = insert / false = no insert).
updateOnMatch	NO_UPDATE REPLACE_ALL	Controls how the existing record should be updated.
matchColumnName1	string	First match column for determining whether an insert or update should occur.
matchColumnName2	string	Second match column for determining whether an insert or update should occur. (optional)
matchOperator	NONE, AND, OR	Controls how the boolean expression involving the match columns is constructed to determine a match between the incoming records and existing records.
optinValue	string	Value of incoming opt-in status data that represents an opt-in status. For example, “1” may represent an opt-in status.
optoutValue	string	Value of incoming opt-out status data that represents an opt-out status. For example, “0” may represent an opt-out status.
defaultPermissionStatus	enum	This value must be specified as either OPTIN or OPTOUT and would be applied to all of the records contained in the API call. If this value is not explicitly specified, then it is set to OPTOUT.
htmlValue	string	Value of incoming preferred email format data. For example, “H” may represent a preference for HTML formatted email.
textValue	string	Value of incoming preferred email format data. For example, “T” may represent a preference for Text formatted email.
rejectRecordIfChannelEmpty	string	String containing comma-separated channel codes that if specified will result in record rejection when the channel address field is null. Channel codes are E, M, P. For example “E,M” would indicate that a record that has a null for Email or Mobile Number value should be rejected.
defaultPermissionStatus	enum	OPTIN, OPTOUT

Campaign Object Attributes

Name	Type	Default Value	Description
id	Long	Null	Id of the campaign.
name	String	Null	Name of the campaign to be created.
folderName	String	Null	Folder name for the campaign to be created.
type	String	EMAIL	Type of the campaign. It can be EMAIL or MOBILE.
description	String	Null	Description of the campaign.
marketingProgram	String	Null	Type of marketing program. The values are defined by the Account Administrator.
marketingStrategy	String	Null	Type of marketing strategy. The values are defined by the Account Administrator.
purpose	String	PROMOTIONAL	Purpose of the campaign. Values: PROMOTIONAL, TRANSACTIONAL
listName	String	Null	Profile list name which contains the audience for this campaign.
filterPaths	List<String>	Null	Paths of filter to be used for this campaign. This can be used to select a group of customers to receive specific messages. Either standard filter or SQL View can be specified.
refiningDataSourcePath	String	Null	Path of additional data sources to be used for this campaign.
proofListPath	String	Null	Before sending the campaign to customers, send it to proof list for testing.
seedListPath	String	Null	Seed lists recipients receive the campaign when it is launched but are excluded from live report.

Name	Type	Default Value	Description
segmentPaths	List<String>	Null	Segmentations are used to divide a list into segments using attributes of a profile extension table or profile list.
supplementary CampaignDataSourcePaths	List<String>	null	Supplementary data sources can be used for inclusions or exclusions of the audience.
supplementary ProofDataSourcePaths	List<String>	Null	Supplementary proof data sources can be used for inclusions or exclusions from proof list.
supplementary SeedDataSourcePaths	List<String>	Null	Supplementary seed data sources can be used for inclusions or exclusions from seed list.
suppressionListPaths	List<String>	Null	This is used to exclude recipients from data sources.
htmlMessagePath	String	Null	HTML message folder and path for the campaign.
textMessagePath	String	Null	Text message folder and path for the campaign.
enableLinkTracking	Boolean	False	Enable to track the links in a campaign.
linkTablePath	String	Null	Link table to be included in this campaign.
attachmentPaths	List<String>	Null	Attachments for the campaign.
enableExternal TrackingParams	Boolean	false	Enable to use third-party web analytics service that use tracking parameters appended to the URLs.
externalTrackingParams	Map<String, String>	Null	Name and value of external tracking parameters.
campaignVariables	Map<String, String>	Null	Name and value of campaign variables which are used as default values for text replacement.
useUTF8	Boolean	False	Use UTF8 encoding for messages.
locale	String	Null	The default recipient locale for this account.

Name	Type	Default Value	Description
trackHTMLOpens	Boolean	False	Enable tracking when each recipient with HTML capability opened the email.
trackConversions	Boolean	False	This applies only when link tracking is enabled for the campaign. This option can be used to record conversion when clicking a link and following through a specific web page.
sendTextIfHTMLUnknown	Boolean	False	If HTML ability is unknown, this option allows sending MIME messages to ensure message text is displayed correctly.
segmentTrackingColumnName	String	Null	Column name for tracking segments.
unsubscribeOption	String	OPTOUT_SINGLE_CLICK	The unsubscribe option for this campaign. Values: NO_OPTOUT_BUTTON, OPTOUT_SINGLE_CLICK, OPTOUT_FORM
unsubscribeFormName	String	Null	Form name to be used for unsubscribing.
autoCloseOption	String	AUTO_CLOSE_X_DAYS_ AFTER_LAST_RESPONSE	Auto close options for this campaign. Values: NO_AUTO_CLOSE, AUTO_CLOSE_ X_DAYS_AFTER_LAUNCH, AUTO_CLOSE_X_DAYS_ AFTER_LAST_RESPONSE, AUTO_CLOSE_ON_DATE
autoCloseValue	String	90	This represents number of days if autoCloseOption is AUTO_CLOSE X_DAYS_AFTER_LAUNCH or AUTO_CLOSE_X_DAYS _AFTER_LAST_RESPONSE. This represents date if autoCloseOption is AUTO_ CLOSE_ON_DATE.
closedCampaignURL	String	Null	The URL to redirect to if a recipient clicks on a link of an already closed campaign.
externalCampaignCode	String	Null	External campaign code.

Name	Type	Default Value	Description
salesForceCampaignId	String	Null	Salesforce campaign id.
scheduleType	String	Null	Valid values are ONCE/NOW
scheduledTime	Date	Null	Date in the format yyyy-mm-dd HH:mm a