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
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.

```json
{
    "recordData": {
        "fieldName": ["RIID_"],
        "records": [
            ["4094326"],
            ["4094327"],
            ["4094328"],
            ["MERGEFAILED: Record 3 = INVALID_PARAMETER: The value ssdcf is not valid for an integer field

        ],
        "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",
            "method": "GET"
        }
    ]
}
```

Sample response in case of failure

```json
{
    "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:
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.

```json
{
    "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_,mobile_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.

```json
{
    "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_,customer_id_&id=9845349498", "method": "GET" },
        { "rel": "mergeListRecipients", "href": "/rest/api/v1/lists/DemoNewsLetterList/members", "method": "POST" }
    ]
}
```

Sample Response in case of failure:

```json
{
    "type": "",
```
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.

```json
[
  {
    "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"
      }
    ]
  }
]"
Sample Response in case of failure:

```json
{
    "type": "",
    "links": [
        {
            "rel": "createProfileExtensionTable",
            "href": "/rest/api/v1/lists/WS_Auto_RILIST/listExtensions",
            "method": "POST"
        }
    ]
}
```
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_SAMPLEx",
        "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_SAMPLEx 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": {
    "fieldName": ["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 ‘fieldName’ 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: RIIID, 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": {
    "fieldName": ["RIID_"],
    "records": [
      ["1761408"],
      ["MERGEFAILED: Record 1 = RECORD DOES NOT MATCH ANY CONTACTS IN THE LIST DemoNewsLetterList\n"],
      ["MERGEFAILED: Record 2 = ERROR: The value xyz is not valid for an integer field\n"],
      ["MERGEFAILED: Record 3 = Field Names length, doesn’t match with Field Values length\n"],
      ["1761409"]
    ],
    "mapTemplateName": null
  }
}
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:

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",
         "method": "GET"
      },
      {
         "rel": "mergeProfileExtensionRecipients",
         "method": "POST"
      },
      {
         "rel": "deleteProfileExtensionRecipientsRIID",
         "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).

```json
{
    "recordData": {
        "fieldNames": ["RIID","SALARY"],
        "records": [
            ["1761409", "239482734"]
        ],
        "mapTemplateName": null,
        "insertOnNoMatch": false,
        "updateOnMatch": null,
        "matchColumn": null,
        "links": [
            {
                "rel": "self",
                "method": "GET"
            },
            {
                "rel": "mergeProfileExtensionRecipients",
```
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).

```json
{
    "recordData": {
        "fieldNames": ["RIID_"],
        "records": [["1761409"]],
        "mapTemplateName": null,
        "insertOnNoMatch": false,
        "updateOnMatch": null,
    }
}
```
"matchColumn": null,
"links": [  
  
  {  "rel": "self",
      "method": "DELETE"
  },  
  
  {  "rel": "mergeProfileExtensionRecipients",
      "method": "POST"
  },  
  
  {  "rel": "retrieveProfileExtensionRecipientsRIID",
      "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",
                        "method": "DELETE"
                    },  
                    
                    {  "rel": "mergeProfileExtensionRecipients",
                        "method": "POST"
                    },  
                    
                    {  "rel": "retrieveProfileExtensionRecipientsRIID",
                        "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:
```json
{
    "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:
```json
{
    "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.*

```json
{
   "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.*

```json
{
   "recordData": {
      "fieldNames": ["PK1", "PK2", "F1", "F2", "PK3"
```
Sample Response in case all primary key fields are not specified in the request payload:

```json
{
  "mapTemplateName": null,
  "recordData": {
    "fieldNames": [
      "PK1",
      "PK2",
      "F1",
      "F2"
    ],
    "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\n",
      "",
      "",
      "",
      ""
    ],
    
    "links": {
      "rel": "self",
      "href": "/rest/api/v1/folders/DemoNewsLetter/suppData/CompositePKSuppTable/members",
      "method": "POST"
    }
  }
}
```
Sample Response in case of failure:

```json
{
  "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"]
}
```
Sample Response in case insertOnMatch or updateOnNoMatch are invalid

**Sample Payload:**
```
{  
  "recordData" : {  
    "fieldName" : ["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" : {  
    "fieldName" : ["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=PK1&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).

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

```plaintext
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).

```json
{
    "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:
```json
[{
   "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:**
```json
{
   "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"
      }
   }]}
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",
}

Optional Data:
{
  "optionalData": [
    {
      "name": "CUSTOM1",
      "value": "value1"
    }
  ]
}
"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>
{
  "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": [

Oracle Responsys REST API Developer’s Guide
"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>"
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",

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/<campaignName>/schedule/<id>",
    "method": "GET"
  },
  {
    "rel": "updateSchedule",
    "href": "/rest/api/v1/campaigns/<campaignName>/schedule/<id>",
    "method": "PUT"
  },
  {
    "rel": "deleteSchedule",
    "href": "/rest/api/v1/campaigns/<campaignName>/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
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": "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"
      }
   ]
}
"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",
      "status": "PAUSED",
      "launchOptions": {
        "proofLaunch": false
      }
    }
  ]
}
"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": null,
      "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"
    }
  ]
}
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:

```json
{
  "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:

```json
{
  "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:

```json
{
  "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"
        }
      ]
    }
  ],
  "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"
        }
      ]
    }
  ]
}
```
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"><p>test document</p><img src="wsrest_cl.images/testcreate-1.png" alt="" /></html>

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"
   }
]
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.

```json
{
   "documentPath": "/contentlibrary/abn/wsrest_cl.htm",
   "content": "\n\n<r\n\n<head>\r
\n<title>\r
\n<body>\r
\n<p>test document</p>\r\n\n<img src="wsrest_cl.images/testcreate-1.png" alt="" />
</body>\r\n\n</html>\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"
      }
   ]
}
```
Sample Response in case of failure:

```json
{
  "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:

```json
{
  "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.*

```json
{
  "documentPath": "/contentlibrary/abn/wsrest_cl.htm",
```
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": []
}
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":  
```
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.

```json
{
  "itemPath": "/contentlibrary/abn/testcreate_50.png",
  "itemData": "iVBORw0KGgoAAAANSUhEUgAAAdwAAAFUCAIAAAABIhoXAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8...
```

Oracle Responsys REST API Developer's Guide
Page 60
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.

```json
{
    "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.

```json
{
    "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:

```json
{
    "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:

```json
{
    "itemPath": "<sourceItemPath>",
    "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 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.

```json
{
    "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:

```json
{
    "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:

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

Sample Response in case of success:

```json
{
  "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:

```json
{
```
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:

```json
{
  "documentPath": "/contentlibrary/abn/wsrest_cl.htm",
  "imageData": [
    {
      "itemPath": "/contentlibrary/abn/testcreate.png",
      "itemData": "<base64 encoded binary string>",
      "links": [
        {
          "rel": "get ContentLibrary Item",
          "href": "/rest/api/v1/clItems/contentlibrary/abn/testcreate.png",
          "method": "GET"
        },
        {
          "rel": "set ContentLibrary Item",
          "href": "/rest/api/v1/clItems/contentlibrary/abn/testcreate.png",
          "method": "POST"
        },
        {
          "rel": "delete ContentLibrary Item",
          "href": "/rest/api/v1/clItems/contentlibrary/abn/testcreate.png",
          "method": "DELETE"
        },
        {
          "rel": "create ContentLibrary Item",
          "href": "/rest/api/v1/clItems",
          "method": "POST"
        }
      ]
    }
  ],
  "links": [
    {
      "rel": "self",
      "href": "/rest/api/v1/clDocImages/contentlibrary/abn/wsrest_cl.htm",
      "method": "GET"
    }
  ]
}
```
Sample Response in case of failure:

```json
{
  "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:**

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

**Error codes:**

<table>
<thead>
<tr>
<th>ERROR_CODE</th>
<th>HTTP response code</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUPLICATE_API_REQUEST</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Duplicate API request</td>
</tr>
<tr>
<td>API_DISABLED_FOR_USER</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>API disabled for user</td>
</tr>
<tr>
<td>INSUFFICIENT_ACCESS</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Insufficient access</td>
</tr>
<tr>
<td>INVALID_USER_NAME_PASSWORD</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Invalid username or password</td>
</tr>
<tr>
<td>INVALID_SESSION_ID</td>
<td>HttpStatus.BAD_REQUEST</td>
<td></td>
</tr>
<tr>
<td>INVALID_SOAP_HEADER</td>
<td>HttpStatus.BAD_REQUEST</td>
<td></td>
</tr>
<tr>
<td>INVALID_NUMBER</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Invalid number</td>
</tr>
<tr>
<td>INVALID_DATE</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Invalid date</td>
</tr>
<tr>
<td>INVALID_PARAMETER</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Invalid Request JSON Body</td>
</tr>
<tr>
<td>INVALID_FIELD_NAME</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Invalid field name</td>
</tr>
<tr>
<td>INVALID_OBJECT</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Invalid object</td>
</tr>
<tr>
<td>PASSWORD_LOCKOUT</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Password locked</td>
</tr>
<tr>
<td>PASSWORD_EXPIRED</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Password expired</td>
</tr>
<tr>
<td>REQUEST_LIMIT_EXCEEDED</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Request limit exceeded</td>
</tr>
<tr>
<td>CAMPAIGN_NOT_FOUND</td>
<td>HttpStatus.NOT_FOUND</td>
<td>Campaign not found</td>
</tr>
<tr>
<td>CAMPAIGN_ALREADY_EXISTS</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign already exists</td>
</tr>
<tr>
<td>RECIPIENT_LIMIT_EXCEEDED</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Recipient limit exceeded</td>
</tr>
<tr>
<td>MAX_ATTACHMENT_SIZE_EXCEEDED</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Attachment size exceeded</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>HTTP response code</td>
<td>Title</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>CAMPAIGN_NOT_LISTENING</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign not listening</td>
</tr>
<tr>
<td>CAMPAIGN_IS_INVALID</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Not a valid campaign</td>
</tr>
<tr>
<td>MOBILE_CAMPAIGN_DISABLED_FOR_USER</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign disabled for user</td>
</tr>
<tr>
<td>FOLDER_NOT_FOUND</td>
<td>HttpStatus.NOT_FOUND</td>
<td>Folder not found</td>
</tr>
<tr>
<td>FOLDER_ALREADY_EXISTS</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Folder already exists</td>
</tr>
<tr>
<td>NO_CAMPAIGNS_IN_THIS_FOLDER</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>No campaigns for this folder</td>
</tr>
<tr>
<td>NOOBJECTS_IN_THIS_FOLDER</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>No objects in this folder</td>
</tr>
<tr>
<td>LIST_NOT_FOUND</td>
<td>HttpStatus.NOT_FOUND</td>
<td>List not found</td>
</tr>
<tr>
<td>LIST_ALREADY_EXISTS</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>List already exists</td>
</tr>
<tr>
<td>TABLE_NOT_FOUND</td>
<td>HttpStatus.NOT_FOUND</td>
<td>Table not found</td>
</tr>
<tr>
<td>CUSTOM_EVENT_NOT_FOUND</td>
<td>HttpStatus.NOT_FOUND</td>
<td>Custom event not found</td>
</tr>
<tr>
<td>RECORD_LIMIT_EXCEEDED</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Record limit exceeded</td>
</tr>
<tr>
<td>RECORD_NOT_FOUND</td>
<td>HttpStatus.NOT_FOUND</td>
<td>Record not found</td>
</tr>
<tr>
<td>OBJECT_NOT_FOUND</td>
<td>HttpStatus.NOT_FOUND</td>
<td>Object not found</td>
</tr>
<tr>
<td>OBJECT_ALREADY_EXISTS</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Object already exists</td>
</tr>
<tr>
<td>OPERATION_NOT_SUPPORTED</td>
<td>HttpStatus.FORBIDDEN</td>
<td>Operation not supported</td>
</tr>
<tr>
<td>MULTIPLE_OBJECTS_FOUND</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Multiple objects found</td>
</tr>
<tr>
<td>DOCUMENT_NOT_FOUND</td>
<td>HttpStatus.NOT_FOUND</td>
<td>Document not found</td>
</tr>
<tr>
<td>DOCUMENT_ALREADY_EXISTS</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Document already exists</td>
</tr>
<tr>
<td>IMAGES_NOT_FOUND</td>
<td>HttpStatus.NOT_FOUND</td>
<td>Images not found</td>
</tr>
<tr>
<td>UNEXPECTED_EXCEPTION</td>
<td>HttpStatus.INTERNAL_SERVER_ERROR</td>
<td>Unexpected exception</td>
</tr>
<tr>
<td>UNRECOVERABLE_EXCEPTION</td>
<td>HttpStatus.INTERNAL_SERVER_ERROR</td>
<td>Unrecoverable exception</td>
</tr>
<tr>
<td>INVALID_AUTHENTICATION_OPTION</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Invalid authentication option</td>
</tr>
<tr>
<td>AUTHENTICATION_FAILED</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Authentication failed</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>HTTP response code</td>
<td>Title</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>CLIENT_CERTIFICATE_EXPIRED</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Client certificate expired</td>
</tr>
<tr>
<td>CLIENT_CERTIFICATE_NOT_YET_VALID</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Client certificate not valid</td>
</tr>
<tr>
<td>CLIENT_CERTIFICATE_NOT_FOUND</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Client certificate not found</td>
</tr>
<tr>
<td>SERVER_CERTIFICATE_EXPIRED</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Server certificate expired</td>
</tr>
<tr>
<td>SERVER_CERTIFICATE_NOT_YET_VALID</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Server certificate not valid</td>
</tr>
<tr>
<td>SERVER_CERTIFICATE_NOT_FOUND</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Server certificate not found</td>
</tr>
<tr>
<td>SERVICE_UNAVAILABLE</td>
<td>HttpStatus.SERVICE_UNAVAILABLE</td>
<td>Service unavailable</td>
</tr>
<tr>
<td>TOKEN_EXPIRED</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Authentication token expired</td>
</tr>
<tr>
<td>METHOD_NOT_SUPPORTED</td>
<td>HttpStatus.METHOD_NOT_ALLOWED</td>
<td>Method not supported</td>
</tr>
<tr>
<td>RESOURCE_NOT_FOUND</td>
<td>HttpStatus.NOT_FOUND</td>
<td>Resource not found</td>
</tr>
<tr>
<td>INVALID_REQUEST_CONTENT</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Invalid request content</td>
</tr>
<tr>
<td>INVALID_TOKEN</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Not a valid token</td>
</tr>
<tr>
<td>PRIVATE_KEY_NOT_FOUND</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Private key not found</td>
</tr>
<tr>
<td>NO_RECIPIENT_FOUND</td>
<td>HttpStatus.NOT_FOUND</td>
<td>No recipient found</td>
</tr>
<tr>
<td>MULTIPLE_RECIPIENTS_FOUND</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Multiple recipients found</td>
</tr>
<tr>
<td>RECIPIENT_STATUS_UNDELIVERABLE</td>
<td>HttpStatus.OK</td>
<td>Recipient status undeliverable</td>
</tr>
<tr>
<td>PROFILE_LIST_NOT_FOUND</td>
<td>HttpStatus.NOT_FOUND</td>
<td>Profile list not found</td>
</tr>
<tr>
<td>PROFILE_LIST_NOT_FOUND_IN_FOLDER</td>
<td>HttpStatus.NOT_FOUND</td>
<td>Profile list not found in folder</td>
</tr>
<tr>
<td>USER_BLOCKED</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>User is blocked</td>
</tr>
<tr>
<td>LOGINS_DISABLED</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Login is disabled</td>
</tr>
<tr>
<td>INVALID_AUTHOPTION</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Not a valid authentication option</td>
</tr>
<tr>
<td>SERVER_CHALLENGES_DO_NOT_MATCH</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Server challenge did not match</td>
</tr>
<tr>
<td>INACTIVE_ACCOUNT</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Account is not active</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>HTTP response code</td>
<td>Title</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>--------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>MAX_CONCURRENT_SESSIONS_EXCEEDED</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Maximum concurrent sessions exceeded</td>
</tr>
<tr>
<td>MAX_LOGIN_FAILURES_EXCEEDED</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Maximum login failure exceeded</td>
</tr>
<tr>
<td>LOGIN_BLOCKED_TEMPORARILY</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Login is blocked temporarily</td>
</tr>
<tr>
<td>ACCOUNT_SUSPENDED</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Account is in suspended state</td>
</tr>
<tr>
<td>LOGIN_BLOCKED_INVALID_IPRANGE</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Not a valid client IP range</td>
</tr>
<tr>
<td>WS_ARG_DISABLED</td>
<td>HttpStatus.UNAUTHORIZED</td>
<td>Web service ARG disabled</td>
</tr>
<tr>
<td>CAMPAIGN_IS_INVALID</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign name is invalid.</td>
</tr>
<tr>
<td>CAMPAIGN_NOT_FOUND</td>
<td>HttpStatus.NOT_FOUND</td>
<td>Campaign not found</td>
</tr>
<tr>
<td>UNABLE_TO_CREATE_CAMPAIGN</td>
<td>HttpStatus.INTERNAL_SERVER_ERROR</td>
<td>Unable to create campaign</td>
</tr>
<tr>
<td>MOBILE_CAMPAIGN_DISABLED_FOR_USER</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign disabled for user</td>
</tr>
<tr>
<td>CAMPAIGN_ALREADY_EXISTS</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign already exists</td>
</tr>
<tr>
<td>PATH_NOT_VALID</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Object path not valid</td>
</tr>
<tr>
<td>CAMPAIGN_SCHEDULE_DUPLICATE</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign is already scheduled for launch</td>
</tr>
<tr>
<td>CURRENT_CAMPAIGN_SCHEDULE_AT_SAME_TIME</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>This Schedule is already scheduled to run at specified time</td>
</tr>
<tr>
<td>CAMPAIGN_LAUNCH_SCHEDULE_DATE_PAST</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign launch date is past</td>
</tr>
<tr>
<td>INVALID_CAMPAIGN_SCHEDULE_TYPE_CHANGE</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign Schedule Type cannot be changed</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>HTTP response code</td>
<td>Title</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>----------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>CAMPAIGN_SCHEDULE_NOT_FOUND</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign Schedule not found</td>
</tr>
<tr>
<td>CAMPAIGN_NAME_NOT_VALID</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign Name is not valid</td>
</tr>
<tr>
<td>INVALID_CAMPAIGN_SCHEDULE_TYPE_CHANGE</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign Schedule Type cannot be changed</td>
</tr>
<tr>
<td>INVALID_CAMPAIGN_SCHEDULE_TYPE</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Invalid Campaign Schedule Type</td>
</tr>
<tr>
<td>CAMPAIGN_LAUNCH_ALREADY_HAPPENED</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign launch has already occurred.</td>
</tr>
<tr>
<td>INVALID_CAMPAIGN_SCHEDULE_TIME</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Invalid Schedule Time.</td>
</tr>
<tr>
<td>CAMPAIGN_IS_INVALID</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign name is invalid.</td>
</tr>
<tr>
<td>CAMPAIGN_NOT_FOUND</td>
<td>HttpStatus.NOT_FOUND</td>
<td>Campaign not found</td>
</tr>
<tr>
<td>UNABLE_TO_CREATE_CAMPAIGN</td>
<td>HttpStatus.INTERNAL_SERVER_ERROR</td>
<td>Unable to create campaign</td>
</tr>
<tr>
<td>MOBILE_CAMPAIGN_DISABLED_FOR_USER</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign disabled for user</td>
</tr>
<tr>
<td>CAMPAIGN_ALREADY_EXISTS</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign already exists</td>
</tr>
<tr>
<td>PATH_NOT_VALID</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Object path not valid.</td>
</tr>
<tr>
<td>CAMPAIGN_SCHEDULE_DUPLICATE</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign is already scheduled for launch</td>
</tr>
<tr>
<td>CURRENT_CAMPAIGN_SCHEDULE_ATSAME_TIME</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>This Schedule is already scheduled to run at specified time</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>HTTP response code</td>
<td>Title</td>
</tr>
<tr>
<td>------------------------------------------------------</td>
<td>--------------------------------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>CAMPAIGN_LAUNCH_SCHEDULE_DATE_PAST</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign launch date is past</td>
</tr>
<tr>
<td>INVALID_CAMPAIGN_SCHEDULE_TYPE_CHANGE</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign Schedule Type cannot be changed</td>
</tr>
<tr>
<td>CAMPAIGN_SCHEDULE_NOT_FOUND</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign Schedule not found</td>
</tr>
<tr>
<td>CAMPAIGN_NAME_NOT_VALID</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign Name is not valid</td>
</tr>
<tr>
<td>INVALID_CAMPAIGN_SCHEDULE_TYPE_CHANGE</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign Schedule Type cannot be changed</td>
</tr>
<tr>
<td>INVALID_CAMPAIGN_SCHEDULE_TYPE</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Invalid Campaign Schedule Type</td>
</tr>
<tr>
<td>CAMPAIGN_LAUNCH_ALREADY_HAPPENED</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Campaign launch has already occurred.</td>
</tr>
<tr>
<td>INVALID_CAMPAIGN_SCHEDULE_TIME</td>
<td>HttpStatus.BAD_REQUEST</td>
<td>Invalid Schedule Time.</td>
</tr>
</tbody>
</table>
# Definitions of Rule Parameters for Merging Members into a Profile List

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>insertOnNoMatch</td>
<td>boolean</td>
<td>Indicates what should be done for records where a match is not found (true = insert / false = no insert).</td>
</tr>
<tr>
<td>updateOnMatch</td>
<td>NO_UPDATE, REPLACE_ALL</td>
<td>Controls how the existing record should be updated.</td>
</tr>
<tr>
<td>matchColumnName1</td>
<td>string</td>
<td>First match column for determining whether an insert or update should occur.</td>
</tr>
<tr>
<td>matchColumnName2</td>
<td>string</td>
<td>Second match column for determining whether an insert or update should occur. (optional)</td>
</tr>
<tr>
<td>matchOperator</td>
<td>NONE, AND, OR</td>
<td>Controls how the boolean expression involving the match columns is constructed to determine a match between the incoming records and existing records.</td>
</tr>
<tr>
<td>optinValue</td>
<td>string</td>
<td>Value of incoming opt-in status data that represents an opt-in status. For example, “1” may represent an opt-in status.</td>
</tr>
<tr>
<td>optoutValue</td>
<td>string</td>
<td>Value of incoming opt-out status data that represents an opt-out status. For example, “0” may represent an opt-out status.</td>
</tr>
<tr>
<td>defaultPermissionStatus</td>
<td>enum</td>
<td>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.</td>
</tr>
<tr>
<td>htmlValue</td>
<td>string</td>
<td>Value of incoming preferred email format data. For example, “H” may represent a preference for HTML formatted email.</td>
</tr>
<tr>
<td>textValue</td>
<td>string</td>
<td>Value of incoming preferred email format data. For example, “T” may represent a preference for Text formatted email.</td>
</tr>
<tr>
<td>rejectRecordIfChannelEmpty</td>
<td>string</td>
<td>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.</td>
</tr>
<tr>
<td>defaultPermissionStatus</td>
<td>enum</td>
<td>OPTIN, OPTOUT</td>
</tr>
</tbody>
</table>
# Campaign Object Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Long</td>
<td>Null</td>
<td>Id of the campaign.</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>Null</td>
<td>Name of the campaign to be created.</td>
</tr>
<tr>
<td>folderName</td>
<td>String</td>
<td>Null</td>
<td>Folder name for the campaign to be created.</td>
</tr>
<tr>
<td>type</td>
<td>String</td>
<td>EMAIL</td>
<td>Type of the campaign. It can be EMAIL or MOBILE.</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>Null</td>
<td>Description of the campaign.</td>
</tr>
<tr>
<td>marketingProgram</td>
<td>String</td>
<td>Null</td>
<td>Type of marketing program. The values are defined by the Account Administrator.</td>
</tr>
<tr>
<td>marketingStrategy</td>
<td>String</td>
<td>Null</td>
<td>Type of marketing strategy. The values are defined by the Account Administrator.</td>
</tr>
<tr>
<td>purpose</td>
<td>String</td>
<td>PROMOTIONAL</td>
<td>Purpose of the campaign. Values: PROMOTIONAL, TRANSACTIONAL</td>
</tr>
<tr>
<td>listName</td>
<td>String</td>
<td>Null</td>
<td>Profile list name which contains the audience for this campaign.</td>
</tr>
<tr>
<td>filterPaths</td>
<td>List&lt;String&gt;</td>
<td>Null</td>
<td>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.</td>
</tr>
<tr>
<td>refiningDataSourcePath</td>
<td>String</td>
<td>Null</td>
<td>Path of additional data sources to be used for this campaign.</td>
</tr>
<tr>
<td>proofListPath</td>
<td>String</td>
<td>Null</td>
<td>Before sending the campaign to customers, send it to proof list for testing.</td>
</tr>
<tr>
<td>seedListPath</td>
<td>String</td>
<td>Null</td>
<td>Seed lists recipients receive the campaign when it is launched but are excluded from live report.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>segmentPaths</td>
<td>List&lt;String&gt;</td>
<td>Null</td>
<td>Segmentations are used to divide a list into segments using attributes of a profile extension table or profile list.</td>
</tr>
<tr>
<td>supplementary CampaignDataSourcePaths</td>
<td>List&lt;String&gt;</td>
<td>null</td>
<td>Supplementary data sources can be used for inclusions or exclusions of the audience.</td>
</tr>
<tr>
<td>supplementary ProofDataSourcePaths</td>
<td>List&lt;String&gt;</td>
<td>Null</td>
<td>Supplementary proof data sources can be used for inclusions or exclusions from proof list.</td>
</tr>
<tr>
<td>supplementary SeedDataSourcePaths</td>
<td>List&lt;String&gt;</td>
<td>Null</td>
<td>Supplementary seed data sources can be used for inclusions or exclusions from seed list.</td>
</tr>
<tr>
<td>suppressionListPaths</td>
<td>List&lt;String&gt;</td>
<td>Null</td>
<td>This is used to exclude recipients from data sources.</td>
</tr>
<tr>
<td>htmlMessagePath</td>
<td>String</td>
<td>Null</td>
<td>HTML message folder and path for the campaign.</td>
</tr>
<tr>
<td>textMessagePath</td>
<td>String</td>
<td>Null</td>
<td>Text message folder and path for the campaign.</td>
</tr>
<tr>
<td>enableLinkTracking</td>
<td>Boolean</td>
<td>False</td>
<td>Enable to track the links in a campaign.</td>
</tr>
<tr>
<td>linkTablePath</td>
<td>String</td>
<td>Null</td>
<td>Link table to be included in this campaign.</td>
</tr>
<tr>
<td>attachmentPaths</td>
<td>List&lt;String&gt;</td>
<td>Null</td>
<td>Attachments for the campaign.</td>
</tr>
<tr>
<td>enableExternal TrackingParams</td>
<td>Boolean</td>
<td>false</td>
<td>Enable to use third-party web analytics service that use tracking parameters appended to the URLs.</td>
</tr>
<tr>
<td>externalTrackingParams</td>
<td>Map&lt;String, String&gt;</td>
<td>Null</td>
<td>Name and value of external tracking parameters.</td>
</tr>
<tr>
<td>campaignVariables</td>
<td>Map&lt;String, String&gt;</td>
<td>Null</td>
<td>Name and value of campaign variables which are used as default values for text replacement.</td>
</tr>
<tr>
<td>useUTF8</td>
<td>Boolean</td>
<td>False</td>
<td>Use UTF8 encoding for messages.</td>
</tr>
<tr>
<td>locale</td>
<td>String</td>
<td>Null</td>
<td>The default recipient locale for this account.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>trackHTMLOpens</td>
<td>Boolean</td>
<td>False</td>
<td>Enable tracking when each recipient with HTML capability opened the email.</td>
</tr>
<tr>
<td>trackConversions</td>
<td>Boolean</td>
<td>False</td>
<td>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.</td>
</tr>
<tr>
<td>sendTextIfHTMLUnknown</td>
<td>Boolean</td>
<td>False</td>
<td>If HTML ability is unknown, this option allows sending MIME messages to ensure message text is displayed correctly.</td>
</tr>
<tr>
<td>segmentTrackingColumnName</td>
<td>String</td>
<td>Null</td>
<td>Column name for tracking segments.</td>
</tr>
<tr>
<td>unsubscribeOption</td>
<td>String</td>
<td>OPTOUT_SINGLE_CLICK</td>
<td>The unsubscribe option for this campaign. Values: NO_OPTOUT_BUTTON, OPTOUT_SINGLECLICK, OPTOUT_FORM</td>
</tr>
<tr>
<td>unsubscribeFormName</td>
<td>String</td>
<td>Null</td>
<td>Form name to be used for unsubscribing.</td>
</tr>
<tr>
<td>autoCloseOption</td>
<td>String</td>
<td>AUTO_CLOSE_X_DAYS_AFTER_LAST_RESPONSE</td>
<td>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</td>
</tr>
<tr>
<td>autoCloseValue</td>
<td>String</td>
<td>90</td>
<td>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.</td>
</tr>
<tr>
<td>closedCampaignURL</td>
<td>String</td>
<td>Null</td>
<td>The URL to redirect to if a recipient clicks on a link of an already closed campaign.</td>
</tr>
<tr>
<td>externalCampaignCode</td>
<td>String</td>
<td>Null</td>
<td>External campaign code.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>---------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>salesForceCampaignId</td>
<td>String</td>
<td>Null</td>
<td>Salesforce campaign id.</td>
</tr>
<tr>
<td>scheduleType</td>
<td>String</td>
<td>Null</td>
<td>Valid values are ONCE/NOW</td>
</tr>
<tr>
<td>scheduledTime</td>
<td>Date</td>
<td>Null</td>
<td>Date in the format yyyy-mm-dd HH:mm a</td>
</tr>
</tbody>
</table>