Oracle Responsys

Developer Guide

REST API v1.3

E65150-22

Important: The REST API v1.3 endpoints in this guide are intended for use with Oracle Responsys 19C and later.

If you need to obtain documentation for the v1.1 REST API endpoints, which *are* still compatible with Oracle Responsys 19C, it is available at the following location: https://community.oracle.com/docs/DOC-1010946

For more details about the differences between the v1.3 and v1.1 REST APIs, please refer to the “Changes and Enhancements” and “Migration Notes” sections of this document.

Contents

What's new

This section describes changes and enhancements to the Oracle Responsys APIs in product updates during the past year.

- Update 19C
- Update 19B
- Update 19A
- Update 18D
- Update 18C

Update 19C

New standard REST API endpoints

You can use the following new endpoints.

- **Publish or Unpublish a Program**: This new API endpoint enables you to publish or unpublish a program. See Publish or Unpublish a Program for details.
- **Create new Profile Extension Table v1.4**: This new API endpoint enables you to set the number of days a profile extension table will exist before expiring. See Create a new profile extension table for details.

Important: On August 19th 2019, Responsys stopped accepting inbound connections using TLS 1.0 and TLS 1.1 protocols. See the Responsys bulletin for more information.

Update 19B

New and updated standard REST API endpoints

You can use the following new and updated endpoints. For more details, see the REST API for Oracle Responsys Marketing Cloud Service reference guide.

- **Organizations REST API endpoints**: New API endpoints that enable you update and retrieve organizational access for campaigns and programs. For more details, see Organizations.
- **Fetch all programs**: A new query parameter, status was added. This new query parameter enables you to filter programs by program status. For example, `https://api5-014.responsys.net/rest/api/v1.3/programs?status=RUNNING`. For more details, see Get all programs.
- **Fetch a campaign**: A new Campaign API endpoint was added. Use the Fetch a Campaign API endpoint to get an existing EMD Email campaign object. For more details, see Get a campaign.

Important: Oracle is updating Responsys application security mechanisms to meet modern security requirements. On August 5th, 2019, Responsys will no longer allow inbound connections using TLS 1.0 and TLS 1.1 protocols. Responsys will continue to support inbound connections using the TLS 1.2 protocol. See the Responsys Bulletin for more information.

Update 19A

New and updated standard REST API endpoints

You can use the following new and updated endpoints. For more details, see the REST API for Oracle Responsys Marketing Cloud Service reference guide.

- **Get all Campaigns**: This API was updated to include MMS and Message Center campaigns as valid campaign types. For more details, see the Get all MMS campaigns and Get all Message Center campaigns sections.
- **Get all Programs**: This new API enables you to get information about the program orchestrations for your Responsys account. For more details, see the Get all programs section.

Update 18D

New and updated standard REST API endpoints

You can use the following new and updated endpoints. For more details about these and other endpoints, see the REST API for Oracle Responsys Marketing Cloud Service reference guide.

- Get primary keys or all field names of a supplemental table: For an existing supplemental table, you can now get the primary key field names or all of the field names.
- Retrieve multiple profile list records in a single request: You can now retrieve multiple profile list records by using a single batch request.
- Delete multiple profile list records in a single request: You can now delete multiple profile list records by using a single batch request.

- Delete multiple profile extension table records in a single request: You can now delete multiple Profile Extension Table (PET) records by using a single batch request.

Documentation enhancements

We have clarified information about how login IP enforcement control affects API user authentication. See Access Controls for more details.

You can perform Email campaign proof launches using the existing APIs for campaign scheduling and Trigger Email Message. Please refer to the Create an Email or Push campaign schedule and Trigger Email Message topics in the the REST API for Oracle Responsys Marketing Cloud Service reference guide for additional details.

Update 18C

There were no additions or changes to the standard REST API in Oracle Responsys 18C.

Support for real-time custom events

Real-time custom events are a special type of custom event that override how Responsys handles enactments when the Enactment Batching feature is enabled. If your account has Enactment Batching enabled and you need to send near-real-time messages, then we highly recommend having your account enabled for the Real-time Events feature. This feature is intended for Responsys customers who use the Mobile App channel. Part of this feature enables you to create real-time custom events. When a real-time custom event is triggered, Responsys handles the enactments in near real-time instead of batching them. This ensures that your customers receive the campaign messages (including Email, SMS, Push, and In-app) without the delay imposed by enactment batching. To have this feature enabled for your account, contact your Oracle

Customer Success Manager. For more information, see the Defining Custom Event Types topic in the *Oracle Responsys Help Center*.

Documentation corrections and enhancements

The request payload example shown for Trigger Push Message incorrectly omitted the `deviceID` and `apiKey` properties when `listType` is `PROFILE`. The properties must be present for each recipient, but the value may be `null`. The topic has also been enhanced with additional information regarding the request and response payloads. See Trigger Push Messages.

Overview

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

About this guide

This guide is organized into three main parts:

- **Getting Started information** - the sections What's new through Login IP enforcement access control (optional) contain the release-specific information and information about how the API works.
- **API endpoint information and request/response examples** - the sections Authenticating through Managing Images of Content Library Documents provide information about using the various REST API calls. We also recommend using the REST API HTML document on docs.oracle.com (https://docs.oracle.com/cloud/latest/marketingcs_gs/OMCED/index.html), where you can view the full request and response parameter descriptions for the API endpoints.

  NOTE: Use the REST HTML document mentioned above if you choose to copy the payload examples and modify them to create your own samples for testing. Copying from this PDF document may result in your sample containing hidden characters or text from the footers (for example, “Page 2”) if you copy text across multiple pages.

- Reference information - the sections Responsys API Data Types through Migration notes provide information about the following topics:

- Data types in the API
- Error messages returned
- Merge rule parameters used in the APIs used for merging data
- Migration notes comparing the changes between API versions

Conventions used in this document

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates terms defined in the text or emphasis.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates URLs, code, text that appears on the screen, or text that you enter.
`{endpoint variable}`	Indicates a variable in the REST endpoint; for example `{campaignName}` should be substituted with the name of the campaign as defined in Responsys.
`<code variable>`	Indicates a variable in the header, request, or response.

About the Oracle Responsys REST API

This release supports several resources, including profile lists, profile extensions, campaigns, programs, organizations, supplemental data, campaign schedules, 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.

Responsys APIs support **only** UTF-8 character encoding. Special characters in request payloads must be UTF-8 encoded. If that is not the case, then an error response is returned.

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 system that hosts the Responsys account, the end points could be one of the following:

   - `login2.responsys.net` (for interact2)
   - `login5.responsys.net` (for interact5)
   - `login.rsys8.net` (for interact8)
   - `login.rsys9.net` (for interact9, when available)

2. Responsys returns a JSON response with a token and an API endpoint URI 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: Design your client API code to create the API endpoint from the URI returned in step 2 **and** the specific path of the desired API. For example, if the URI from step 2 were `<ENDPOINT_URI>`, then the API endpoint to get all profile lists for an account would be `https://<ENDPOINT_URI>/rest/api/v1.3/lists`.

4. If the API request is successfully processed, Responsys returns a specific JSON response according to the specification of the processed API. Otherwise, Responsys returns an error payload 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 a fixed period of time (two hours).

API Call Processing

Web Services API calls are processed synchronously. For most calls, you should receive a response shortly after Responsys finishes processing the call. However, some of the API calls trigger system actions that are performed after the system sends the positive response back to the API caller. If those actions fail for some reason, you may have received a positive response for the API call, but the failure for subsequent actions would be recorded elsewhere.

Examples:

- API calls that trigger messages requiring personalization. Responsys processes personalization asynchronously after the API call has returned a positive response. If the personalization fails, then you may receive a positive API response, even though the message was not sent to the recipient.
- Trigger custom event API calls. These calls do not actually send the email or mobile messages. The triggerCustomEvent API call merely sends a group of recipients (that is, enactments) to a Responsys Program. The Program subsequently uses its own logic to determine if those enactments will be used by campaigns within that Program. The campaigns ultimately send the email or mobile messages.

How Enactment Batching Affects Processing

Oracle Responsys enables cross-channel orchestrations with Email, SMS, and Push. **If you plan to use the trigger custom event API call with cross-channel marketing programs, please review this section first.**

Responsys requires the Enactment Batching feature to be enabled when using trigger custom event with Mobile App campaigns in Program. Otherwise, the Mobile App campaign events in the program will not be processed. However, there are some tradeoffs to consider before enabling the feature. When an account has Enactment Batching enabled, triggering a custom event cannot be used to perform near-real-time processing for **any** campaign type. Responsys will batch enactments together into a single enactment group before entering the enactments into a program. This results in **at least** a 10-minute delay between custom event triggering and entry into a Program.

If your account has Enactment Batching enabled and you need to send near-real-time messages, such as event reactions, then we highly recommend having your account enabled for the Real-time Events feature. This feature is intended for Responsys customers who use the Mobile App channel. Part of this feature enables you to create real-time custom events. Real-time custom events are a special type of custom event that override how Responsys handles enactments when the Enactment Batching feature is enabled. When a real-time custom event is triggered, Responsys handles the enactments in near real-time instead of batching them. This ensures that your customers receive the campaign messages (including Email, SMS, Push, and In-app) without the delay imposed by enactment batching. To have this feature enabled for your account, contact your Oracle Customer Success Manager. For more information, see the Defining Custom Event Types topic in the *Oracle Responsys Help Center*.

Alternatively, you can send near-real-time messages by using merge-trigger API calls (or, for Mobile Push, perform a merge call and then follow it by a trigger push messages call). You can then enter the recipients into a Program orchestration, using schedule filter or other methods. Please contact your Customer Success Manager (CSM) for additional assistance or information.

Access Controls

This section presents Responsys capabilities for controlling user access to APIs.

Organizational access control

When Organizational Access Control is enabled for a Responsys account, it will be enforced for all users in that account, including API users. This means that the organizational units to which the user is assigned will limit the API user’s access to Responsys objects. Similarly, objects created through the API will inherit organizational membership of the API user. For the API user to access all objects within the account, that API user should be assigned to the Root node in the organizational hierarchy.

Organizational Access Control is configured through "Account | Manage Users | Organization Assignment". Please contact your Responsys account administrator to set up access control.

Functional access control

API user’s access level to a specific object is determined by functional roles that are assigned to that user. Functional Access Control and Organizational Access Control work together. Organizational Access Control determines whether the API user has access to a particular object, and Functional Access Control determines what operations the user can perform with that object. For example, a user may have access to the profile extension object but cannot add or update data in it.

Best practice recommendation is to use a dedicated user for API operations. API user should be assigned one or more functional roles (Campaign Web Services Manager, Folder Web Services Manager, Table Web Services Manager, Content Web Services Manager, or List Web Services Manager) to ensure adequate access level to the appropriate set of objects.

Functional Access Control is configured through "Account | Manage Users | Role Assignment". Please contact your Responsys account administrator to setup access controls.

Login IP enforcement access control (optional)

Oracle Responsys enables customers to limit login access based on their defined range (s) of authorized login IP addresses. Login IP restrictions are optional and require the customer to work with Oracle Responsys Support to set this up.

If login IP restrictions are in effect for your account, the system immediately denies any login attempts initiated outside of your authorized ranges of login IP addresses. These restrictions apply to the API user as well as to users logging in to the Responsys user interface. For customers with login IP restrictions in effect, Oracle recommends that you ensure that the IP of the gateway server sending the API requests is one of the authorized login IP addresses.

To view the IP access list settings, a Responsys account administrator can log in and to go “Account | View login IP restrictions”.

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 in the response. You must use these **authToken** and **endPoint** values for any subsequent REST API request.

Transport Layer Security Support

Ensure that your client systems use Transport Layer Security (TLS) version 1.2.

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/v1.3/auth/token
```

Request Method:

```
POST
```

Request Header

```
Content-Type: application/x-www-form-urlencoded
```

Request Parameters

Send in message body, x-www-form-urlencoded:

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

Sample Login Request:

URL:

```
/rest/api/v1.3/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. Logging in with certificates is based on the use of a digital certificate in accordance with the X.509 standard for public key infrastructure (PKI).

Before you can use this type of login in a client application, the Oracle CX Audience account administrator must log in to the Oracle CX Audience user interface and navigate to the admin console for managing users. From the console, the admin must enable the API user to use certificates, upload a digital certificate (client user public key), and download the Oracle CX Audience API server digital certificate (server public key).

Performing this type of authentication requires two REST API calls, as described in the steps below. To see this illustrated in an example script, refer to Authentication with Certificates.

1. Authenticate server by sending the following REST request.

   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/v1.3/auth/token
   ```

Request Method:

```
POST
```

Request Header

```
Content-Type: application/x-www-form-urlencoded
```

Request Parameters

Send in the Request Body in x-www-form-urlencoded format:

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

For REST, the `client_challenge` must be a plain random number and/or text string of your choice, which you must then convert to byte and then Base64 encode.

2. You should receive the following response from the server. Decrypt (using the RSA algorithm) the encrypted `clientChallenge` using server certificate's public key (which you should have downloaded and stored on your system). If they match, proceed to the next step. If they don't match, please do not proceed with steps 3 and 4 in this section. Instead, contact Oracle Support.

Response:

```
{
 "authToken" : "<TEMP_AUTH_TOKEN>",
 "serverChallenge" : "<BASE_64_ENCODED_SERVER_CHALLENGE>",
 "clientChallenge" : "<ENCRYPTED_AND_THEN_BASE_64_ENCODED_
CLIENT_CHALLENGE>"
}
```

3. Log in with username and certificate using the following REST request (do the encryption using the public key from the Responsys server certificate):

   Service URL:

   `/rest/api/v1.3/auth/token`

   Request Method:

   `POST`

   Request Header

   ```
   Authorization=<TEMP_AUTH_TOKEN>   (this is obtained from the
   response in step 2 above)
   Content-Type: application/x-www-form-urlencoded
   ```

   Request Parameters:

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

```
PUBLIC_KEY>
```

4. You should receive the following response from the server. You can use this authentication token and endpoint until the token expires (2 hours from issue time) or until you refresh the token and receive a new one, as described in the next section.

   Response:

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

Refresh token

In the REST API, the authorization token is stateless and it always expires after **two hours**. However, you can refresh the existing token before it expires. If you refresh the token, the system generates a **new** token from the existing valid one, so that you will not need to re-authenticate. The same token used previously is **not** returned.

Service URL:

```
/rest/api/v1.3/auth/token
```

Request Method:

```
POST
```

Request Parameters:

```
auth_type=token
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Response:

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

Get Throttling Limits

Responsys monitors and throttles the frequency of API requests that are submitted from each Oracle Responsys account. This is to ensure that the best possible level of service is offered to API clients in a shared environment.

You can use this API to obtain a list of API throttling limits for key interfaces for your Responsys account.

Service URL:

`/rest/api/ratelimit`

IMPORTANT: Unlike the other REST API calls, the service URL for Get Throttling limit does not include a version number. This is because it is handled directly by the Oracle API Gateway server. If you include a version number, you will receive an error response (`HTTP Status 404 - Not Found`).

Request Method:

```
GET
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request Body:

Not applicable

Response:

RESPONSE NOTES: A successful response returns your Responsys account-specific throttling limits of all APIs configured on the gateway server, and it may include some APIs that are not enabled for your account. If your account is not configured for throttling for a specific API, the gateway server returns the default throttle limit for that API.

- A successful response will return the following for each REST API for which throttling limits apply:
    - `limit`: Number of requests allowed per minute for the API.
    - `api`: System name of the API for which the limit applies.
    - `resource_path`: The resource path of the API. For example, for Merge Trigger Email, it would show `/campaigns/{campaignName}/email`.
    - `verb`: The API method (GET, POST, PUT, or DELETE).
    - `description`: Text description of the API.
- Common error responses include the following:
    - 404: HTTP Status 404 – Not Found. One common cause for this is when “v1.3” is included in the endpoint path. Ensure that you are using `/rest/api/ratelimit`.
    - 500: UNEXPECTED_EXCEPTION (“Not a valid authentication token”). You must use a valid authentication token in your request. Try logging in again and use the authToken returned in the successful login response.

Successful Response Example

Note: The values shown in the following example are from a test environment and represent a partial sample of the response body. **The values in the response you receive for your account will differ.**

```
[
    {
        "limit": "10",
        "api": "login",
        "resource_path": "auth/token",
        "verb": "POST",
        "description": "Login"
    },
    {
        "limit": "100",
        "api": "retrieveProfileLists",
        "resource_path": "lists",
        "verb": "GET",
        "description": "Fetch All Profile Lists"
    },
.
.
.
    {
        "limit": "1000",
        "api": "mergeListRecipients",
        "resource_path": "lists/{listName}/members",
        "verb": "POST",
        "description": "Merge List Recipients"
    },
    {
        "limit": "100",
        "api": "retrieveListRecipients",
        "resource_path": "lists/{listName}/members",
        "verb": "GET",
        "description": "Retrieve List Recipient using query
attribute"
    },
.
.
.
```

```
    {
        "limit": "1000",
        "api": "HaMergeListRecipients",
        "resource_path": "lists/{listName}/members",
        "verb": "POST",
        "description": "Merge List Recipients"
    }
]
```

More about throttling limits for the Responsys Web Services API:

Depending on the type of API function, a specific frequency rate limit is imposed based on an account's total number of requests made per minute for that function. For example, the API function for triggering email messages can be called more times per minute than the API function for launching a campaign. By default, the throttling limit for high volume API functions (for example, triggering email messages or merging records into a profile list) is set to 200 requests per minute.

When an account exceeds its allowable frequency rate limit for an API request, you see the error code `API_LIMIT_EXCEEDED` and this message “`You exceeded your allowable limit to call the <function_name> API function. Please try again in a minute.`” On the other hand, if a specific user of an account is blocked from using selected API functions, the user sees the error code `API_BLOCKED` with this message: “`The <function_name> is currently not available to this user. Please contact tech support.`”

REST API v1.3 resources

Managing Profile Lists

All profile lists in an account can be retrieved. Also, members of a list can be retrieved, and new members can be added to a list or attribute values of existing members can be updated. Finally, a member of a profile list can be deleted using its RIID.

Retrieving all profile lists for an account

Use this interface to retrieve all profile lists for an account.

Service URL:

```
/rest/api/v1.3/lists
```

Request Method:

```
GET
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request Body:

None

Sample Response in case of success:

RESPONSE NOTE: The response is a collection of Profile Lists. Each of the individual objects in the collection represents a Profile List Object.

```
[
  {
    "name": "DemoProfileList",
    "fields": [
      {
        "fieldName": "RIID_",
        "fieldType": "INTEGER"
      },
      {
        "fieldName": "CREATED_SOURCE_IP_",
        "fieldType": "STR255"
      },
      {
        "fieldName": "CUSTOMER_ID_",
        "fieldType": "STR255"
      },
      {
        "fieldName": "EMAIL_ADDRESS_",
        "fieldType": "STR500"
      },
      {
        "fieldName": "EMAIL_DOMAIN_",
        "fieldType": "STR255"
      },
      {
        "fieldName": "EMAIL_ISP_",
        "fieldType": "STR255"
      },
      {
        "fieldName": "EMAIL_FORMAT_",
        "fieldType": "CHAR"
      },
      {
        "fieldName": "EMAIL_PERMISSION_STATUS_",
        "fieldType": "CHAR"
      },
      {
        "fieldName": "EMAIL_DELIVERABILITY_STATUS_",
        "fieldType": "CHAR"
      },
      {
        "fieldName": "EMAIL_PERMISSION_REASON_",
        "fieldType": "STR255"
      },
      {
        "fieldName": "EMAIL_MD5_HASH_",
        "fieldType": "STR50"
      },
      {
        "fieldName": "EMAIL_SHA256_HASH_",
        "fieldType": "STR100"
      },
      {
        "fieldName": "MOBILE_NUMBER_",
        "fieldType": "STR50"
      },
      {
        "fieldName": "MOBILE_COUNTRY_",
        "fieldType": "STR25"
      },
      {
        "fieldName": "MOBILE_PERMISSION_STATUS_",
        "fieldType": "CHAR"
      },
      {
        "fieldName": "MOBILE_DELIVERABILITY_STATUS_",
        "fieldType": "CHAR"
      },
      {
        "fieldName": "MOBILE_PERMISSION_REASON_",
        "fieldType": "STR255"
      },
      {
        "fieldName": "POSTAL_STREET_1_",
        "fieldType": "STR255"
      },
      {
        "fieldName": "POSTAL_STREET_2_",
        "fieldType": "STR255"
      },
      {
        "fieldName": "CITY_",
        "fieldType": "STR50"
      },
      {
        "fieldName": "STATE_",
        "fieldType": "STR50"
      },
      {
        "fieldName": "POSTAL_CODE_",
        "fieldType": "STR25"
      },
      {
        "fieldName": "COUNTRY_",
        "fieldType": "STR50"
      },
      {
        "fieldName": "POSTAL_PERMISSION_STATUS_",
        "fieldType": "CHAR"
      },
      {
        "fieldName": "POSTAL_DELIVERABILITY_STATUS_",
        "fieldType": "CHAR"
      },
      {
        "fieldName": "POSTAL_PERMISSION_REASON_",
        "fieldType": "STR255"
      },
      {
        "fieldName": "CREATED_DATE_",
        "fieldType": "TIMESTAMP"
      },
      {
        "fieldName": "MODIFIED_DATE_",
        "fieldType": "TIMESTAMP"
      }
    ]
  },
  {
    "name": "DemoList",
    "fields": [
      {
        "fieldName": "RIID_",
        "fieldType": "INTEGER"
      },
      ...
         {
        "fieldName": "MODIFIED_DATE_",
        "fieldType": "TIMESTAMP"
      },
      {
        "fieldName": "FIRST_NAME",
        "fieldType": "STR100"
      },
      {
        "fieldName": "LAST_NAME",
        "fieldType": "STR100"
      }
    ]
  }
]
```

Sample Response in case of failure:

```
{
   "type": "",
   "title": "List not found",
   "errorCode": "LIST_NOT_FOUND",
   "detail": "Account : datateam : does not have any profile
lists.",
   "errorDetails": []
}
```

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, 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 in Definitions of Rule Parameters for Merging Members into a Profile List.

REQUEST NOTES:

- Up to 200 members can be handled per a single request.
- The `matchColumnName` attributes can have the following possible values: `RIID_`, `CUSTOMER_ID_`, `EMAIL_ADDRESS_`, `MOBILE_NUMBER_`, `EMAIL_MD5_HASH_`, `EMAIL_SHA256_HASH_`
- Limitations:
  - A combination of either of the email HASH keys or a combination of `EMAIL_ADDRESS` with either of the email HASH keys cannot be given as `matchColumnNames` at the same time.
  - When either of the email HASH keys exists as a match column, then only updates are possible. Set the corresponding value in `insertOnNoMatch` to false when using these columns as match columns.

Service URL:

`/rest/api/v1.3/lists/{listName}/members`

Request Method:

```
POST
```

Request Header:

```
Authorization=<AUTH_TOKEN>

Content-Type=application/json
```

Request 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:

RESPONSE NOTES:

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

```
{
    "recordData": {
        "fieldNames": [
            "RIID_"
        ],
        "records": [
            [
                "2047888987"
            ],
            [
                "4094327"
            ],
            [
                "2047889007"
            ],
            [
                "MERGEFAILED: Record 3 = INVALID_PARAMETER: The
value ssdcf is not valid for an integer field\n\n\r\n"
            ]
        ],
        "mapTemplateName": null
    },
    "mergeRule": {
        "textValue": "T",
        "insertOnNoMatch": true,
        "updateOnMatch": "REPLACE_ALL",
        "htmlValue": "H",
        "optinValue": "I",
        "matchColumnName1": "RIID_",
        "matchColumnName2": null,
        "matchOperator": "NONE",
        "optoutValue": "O",
        "rejectRecordIfChannelEmpty": null,
        "defaultPermissionStatus": "OPTIN",
        "matchColumnName3": null
    },
    "links": [
        {
            "rel": "self",
            "href": "/rest/api/v1.3/lists/Auto Filters List HA
enabled/members",
            "method": "POST"
        },
        {
            "rel": "retrieveListRecipientsRIID",
            "href": "/rest/api/v1.3/lists/Auto Filters List HA
enabled/members/<riid>",
            "method": "GET"
        },
        {
            "rel": "deleteListRecipientsRIID",
            "href": "/rest/api/v1.3/lists/Auto Filters List HA
enabled/members/<riid>",
            "method": "DELETE"
        }
    ]
```

```
}
```

Sample Response in case of failure:

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

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).

REQUEST NOTES:

- 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/v1.3/lists/{listName}/members/{riid}`

Request Method:

`GET`

Request Header:

`Authorization=<AUTH_TOKEN>`

Request parameters:

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

Request Body:

None

Sample Response in case of success:

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

```
{
  "recordData": {
    "fieldNames": [
      "RIID_",
      "EMAIL_ADDRESS_",
      "CUSTOMER_ID_"
    ],
    "records": [
      [
        "4094330",
        "ab.na@gmail.com",
        null
      ],
      [
        "4094326",
        "ab.cd@gmail.com",
        null
      ]
    ],
    "mapTemplateName": null
  },
  "mergeRule": {
    "textValue": null,
    "insertOnNoMatch": false,
    "updateOnMatch": null,
    "matchOperator": null,
    "matchColumnName3": null,
    "matchColumnName1": null,
    "matchColumnName2": null,
    "optinValue": null,
    "optoutValue": null,
    "rejectRecordIfChannelEmpty": null,
    "htmlValue": null,
    "defaultPermissionStatus": null
  },
  "links": [
    {
      "rel": "self",
      "href":
"/rest/api/v1.3/lists/DemoNewsLetterList/members?qa=m&amp;fs=riid_,
email_address_,customer_id_&amp;id=9845349498",
      "method": "GET"
    },
    {
      "rel": "mergeListRecipients",
      "href": "/rest/api/v1.3/lists/DemoNewsLetterList/members",
      "method": "POST"
    },
    {
      "rel": "deleteListRecipientsRIID",
      "href":
"/rest/api/v1.3/lists/DemoNewsLetterList/members/<riid>",
      "method": "DELETE"
    }
  ]
}
```

Sample Response in case of failure:

```
{
   "type": "",
   "title": "Record not found",
   "errorCode": "RECORD_NOT_FOUND",
   "detail": "No records found in the 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.

REQUEST NOTES:

- 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/v1.3/lists/{listName}/members`

Request Method:

`GET`

Request Header:

`Authorization=<AUTH_TOKEN>`

Request parameters:

- `qa – Query Attribute. Can be one of the following values:`
  - `‘r’ - RIID`
  - `‘e’ - EMAIL_ADDRESS`
  - `‘c’ - CUSTOMER_ID`
  - `‘m’ - MOBILE_NUMBER`
- `id – ID corresponding to the query attribute`
- `fs – Comma separated field list or ‘all’`

Request Body:

None

Sample Response in case of success:

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

```
{
   "recordData":    {
      "fieldNames":       [
         "RIID_",
         "EMAIL_ADDRESS_",
         "CUSTOMER_ID_"
      ],
      "records":       [
         [
            "4094330",
            "ab.na@gmail.com",
            null
         ],
         [
            "4094326",
            "ab.cd@gmail.com",
            null
         ]
      ],
      "mapTemplateName": null
   },
   "mergeRule":    {
      "textValue": null,
      "insertOnNoMatch": false,
      "updateOnMatch": null,
      "matchOperator": null,
      "matchColumnName3": null,
      "matchColumnName1": null,
      "matchColumnName2": null,
      "optinValue": null,
      "optoutValue": null,
      "rejectRecordIfChannelEmpty": null,
      "htmlValue": null,
      "defaultPermissionStatus": null
   },
   "links":    [
      {
         "rel": "self",
         "href":
"/rest/api/v1.3/lists/DemoNewsLetterList/members?qa=m&fs=riid_,emai
l_address_,customer_id_&id=9845349498",
         "method": "GET"
      },
      {
         "rel": "mergeListRecipients",
         "href": "/rest/api/v1.3/lists/DemoNewsLetterList/members",
         "method": "POST"
      }
   ]
}
```

Sample Response in case of failure:

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

Retrieve multiple profile list records in a single request

Your client application can retrieve multiple profile list records in a single batch request by using the query attribute `action=get` and by passing a list of IDs.

REQUEST NOTES:

- For this request to work, your endpoint must include the query attribute `action=get` (as shown in the Service URL below).
- Note that the request method for this API is `POST`, because you pass the query attribute and the IDs to retrieve in the request body.
- You can send up to 200 IDs in the request body.

- 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/v1.3/lists/{listName}/members`**`?action=get`**

Request Method:

`POST`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Request Body:

Request attributes:

`queryAttribute` - Specifies the query attribute that you are sending. It can be one of the following values:

- `r` - RIID
- `e` - Email address
- `c` - Customer ID
- `m` - Mobile number

`fieldList` - Comma-separated list of the fields you want from the records you are retrieving. Set this value to `all` to get all Profile List fields for each record returned. Each field name must be 150 characters or fewer.

`ids` - Array containing the identifier values corresponding to the query attribute. For example, if you specify `c`, the system expects the `ids` array to contain the customer ID values for the records you want to retrieve. You can pass up to 200 IDs in the request, and each ID must be 500 characters or fewer.

Sample request endpoint and body:

The following sample endpoint and request are requesting two records from the profile list **MyExampleProfileList** for people whose email addresses are **recipient1@example.com** and **recipient2@example.com**. The `fieldList` lists the fields that will be returned for each recipient in the response.

Sample request endpoint:

```
POST /rest/api/v1.3/lists/MyExampleProfileList/members?action=get
```

Sample request body:

```
{
  "queryAttribute" : "e",
  "fieldList" : ["RIID_","EMAIL_ADDRESS_","CUSTOMER_ID_","POSTAL_
STREET_2_","POSTAL_STREET_1_","CITY_","STATE_","POSTAL_CODE_
","COUNTRY_"],
  "ids" : ["recipient1@example.com", "recipient2@example.com"]
}
```

Sample Response in case of success:

RESPONSE NOTES:

- For each match, 10 records maximum will be returned. For example, if there are 11 records for a given mobile number, only the most recent 10 records will be returned.
- If the system cannot find a record for a given ID, it will not return either a result or an error message. The system only returns an error message if it cannot find any records for *all* of the IDs passed.
- Other attributes in the response like `mapTemplateName` and `mergeRule` will have default values of null/false.

```
HTTPS response status code: 200 OK

{
  "recordData":
    { "fieldNames": ["RIID_","EMAIL_ADDRESS_","CUSTOMER_ID_
","POSTAL_STREET_2_","POSTAL_STREET_1_","CITY_","STATE_","POSTAL_
CODE_","COUNTRY_"],
      "records": [
        [ "63036487","recipient1@example.com","481",null,"1427 CLAY
ST","San Francisco","CA",null,"USA" ],
        [ "63027514","recipient2@example.com","818",null,"1993
O'Shaughnessy Blvd","San Francisco","CA","94131","USA" ]
      ],
"mapTemplateName": null },

  "mergeRule":
    {"textValue": null,
     "htmlValue": null,
     "optinValue": null,
     "insertOnNoMatch": false,
     "updateOnMatch": null,
     "matchColumnName1": null,
     "matchColumnName2": null,
     "matchOperator": null,
     "optoutValue": null,
     "rejectRecordIfChannelEmpty": null,
     "defaultPermissionStatus": null,
     "matchColumnName3": null },
  "links": [
    { "rel": "self",
      "href":
"/rest/api/v1.3/lists/MyExampleProfileList/members?action=get",
      "method": "POST" },
    { "rel": "deleteMultipleListRecipients",
      "href":
"/rest/api/v1.3/lists/MyExampleProfileList/members?action=delete",
      "method": "POST" },
    { "rel": "mergeListRecipients",
      "href": "/rest/api/v1.3/lists/MyExampleProfileList/members",
      "method": "POST" }
  ]
}
```

Troubleshooting error responses

The system returns error responses when:

No records found for *all* of the IDs: For example, if you sent 30 IDs and no records were found for any of them, you would receive this error. A `404 not found` error is returned with the following error response body:

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

List not found: The system could not find the Profile List specified in the request endpoint. A `404 not found error` is returned with the following error response body:

```
{
  "type": "",
  "title": "List not found",
  "errorCode": "LIST_NOT_FOUND",
  "detail": "MyOtherExampleProfileList List Not Found",
  "errorDetails": []
}
```

ID type does not match the field type: A `400 bad request` error is returned with the following error response body:

```
{
  "type": "",
  "title": "Invalid request parameters",
  "errorCode": "INVALID_PARAMETER",
  "detail": "The value recipient1@example.com is not valid for an
integer field",
  "errorDetails": []
}
```

queryAttribute is null or empty: A `400 bad request` error is returned with the following error response body (`queryAttribute` is called "`QueryColumn`" in the error message):

```
{
  "type": "",
  "title": "Invalid request parameters",
  "errorCode": "INVALID_PARAMETER",
  "detail": "QueryColumn is null OR empty",
  "errorDetails": []
}
```

fieldList array is null or empty: : A `400 bad request` error is returned with the following error response body:

```
{
  "type": "",
  "title": "Invalid request parameters",
  "errorCode": "INVALID_PARAMETER",
  "detail": "fieldList is empty",
  "errorDetails": []
}
```

No values provided in the ids array: : A `400 bad request` error is returned with the following error response body:

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

One or more ids array value is more than 500 characters: For example, if you used email address as the `queryAttribute` and one of the email addresses was more than 500 characters, then this error would occur. A `400 bad request` error is returned with the following error response body:

```
{
  "type": "",
  "title": "Invalid request parameters",
  "errorCode": "INVALID_PARAMETER",
  "detail": "IdToRetrieve exceeds limit of 500 characters.",
  "errorDetails": []
}
```

One or more of the field names sent in the fieldList array is more than 150 characters: A `400 bad request` error is returned with the following error response body:

```
{
  "type": "",
  "title": "Invalid request parameters",
  "errorCode": "INVALID_PARAMETER",
  "detail": "FieldList exceeds limit of 150 characters.",
  "errorDetails": []
}
```

The fieldList array contains one or more invalid field names: A 400 bad request error is returned with the following error response body (where the invalid field names sent are returned in the square brackets). In the following example, “RIID” is flagged as invalid because the field name in Responsys is “RIID_”:

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

The `queryAttribute` value is not valid.: A `400 bad request` error is returned with the following error response body:

```
{
  "type": "",
  "title": "Invalid request parameters",
  "errorCode": "INVALID_PARAMETER",
  "detail": "Query Attribute Must be either r, e, c or m",
  "errorDetails": []
}
```

Record limit exceeded: The 200-record limit is exceeded (that is, more than 200 query ID values are sent). A `400 bad request` error is returned with the following error response body.

```
{
  "type": "",
  "title": "Record limit exceeded",
  "errorCode": "RECORD_LIMIT_EXCEEDED",
  "detail": "Record limit exceeded, maximum of 200 records are
allowed per each api call",
  "errorDetails": []
}
}
```

API limit exceed: When the client application exceeds the throttling limit for this API, a `401 Unauthorized` error is returned with the following error response body:

```
{
  "type": "",
  "title": "",
  "errorCode": "API_LIMIT_EXCEEDED",
  "detail": "",
  "errorDetails": []
}
```

Delete profile list recipients based on RIID

Use this interface to delete a profile list member by specifying the RIID for that member.

Service URL:

`/rest/api/v1.3/lists/{listName}/members/{riid}`

Request Method:

`DELETE`

Request Header:

`Authorization=<AUTH_TOKEN>`

Request Body:

None

Sample Response in case of success:

REQUEST NOTES:

- 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 for some reason, 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.
- Other attributes in the response (`mapTemplateName`, `mergeRule`) will have default values (null/false).

```
{{
   "recordData":    {
      "fieldNames": ["RIID_"],
      "records": [["1561"]],
      "mapTemplateName": null
   },
   "mergeRule":    {
      "textValue": null,
      "matchColumnName3": null,
      "matchColumnName1": null,
      "matchColumnName2": null,
      "rejectRecordIfChannelEmpty": null,
      "defaultPermissionStatus": null,
      "updateOnMatch": null,
      "matchOperator": null,
      "optinValue": null,
      "optoutValue": null,
      "htmlValue": null,
      "insertOnNoMatch": false
   },
   "links":    [
            {
         "rel": "self",
         "href":
"/rest/api/v1.3/lists/DemoProfileList/members/1561",
         "method": "DELETE"
      },
            {
         "rel": "mergeListRecipients",
         "href": "/rest/api/v1.3/lists/DemoProfileList/members",
         "method": "POST"
      },
            {
         "rel": "retrieveListRecipientsRIID",
         "href":
"/rest/api/v1.3/lists/DemoProfileList/members/<riid>",
         "method": "GET"
      }
   ]
}
```

Sample Response in case recipient is not found:

```
{
  "type": "",
  "title": "No recipient found",
  "errorCode": "NO_RECIPIENT_FOUND",
  "detail": "Record not Found in the List.",
  "errorDetails": []
}
```

Sample Response in case of failure:

```
 {
   "recordData":    {
      "fieldNames": ["RIID_"],
      "records": [["DELETEFAILED: ERROR: The value abdce is not
valid for an integer field"]],
      "mapTemplateName": null
   },
   "mergeRule":    {
      "textValue": null,
      "matchColumnName3": null,
      "matchColumnName1": null,
      "matchColumnName2": null,
      "rejectRecordIfChannelEmpty": null,
      "defaultPermissionStatus": null,
      "updateOnMatch": null,
      "matchOperator": null,
      "optinValue": null,
      "optoutValue": null,
      "htmlValue": null,
      "insertOnNoMatch": false
   },
   "links":    [
            {
         "rel": "self",
         "href":
"/rest/api/v1.3/lists/DemoProfileList/members/abcde",
         "method": "DELETE"
      },
            {
         "rel": "mergeListRecipients",
         "href": "/rest/api/v1.3/lists/DemoProfileList/members",
         "method": "POST"
      },
            {
         "rel": "retrieveListRecipientsRIID",
         "href":
"/rest/api/v1.3/lists/DemoProfileList/members/<riid>",
         "method": "GET"
      }
   ]
}
```

Delete multiple profile list records in a single request

Your client application can delete multiple profile list records in a single batch request by using the query attribute `action=delete` and by passing a list of IDs.

REQUEST NOTES:

- For this request to work, your endpoint must include the query attribute `action=delete` (as shown in the Service URL below).
- Note that the request method for this API is `POST`, because you pass the query attribute and the IDs to delete in the request body.
- You can send up to 200 IDs in the request body.

Service URL:

`/rest/api/v1.3/lists/{listName}/members`**`?action=delete`**

Request Method:

`POST`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Request Body:

Request attributes:

`queryAttribute` - Specifies the query attribute to use for locating the records that you want to delete. It can be one of the following values:

- `r` - RIID
- `e` - Email address
- `c` - Customer ID
- `m` - Mobile number

`ids` - Array containing the identifier values corresponding to the query attribute. For example, if you specify `c`, the system expects the `ids` array to contain the customer ID values for the records you want to delete. You can pass up to 200 IDs in the request, and each ID must be 500 characters or fewer.

Sample request endpoint and body:

When the server receives the following sample endpoint and request, the system will attempt to delete two records from the profile list **MyExampleProfileList** for people whose email addresses are **recipient1@example.com** and **recipient2@example.com**.

Sample request endpoint:

```
POST
/rest/api/v1.3/lists/MyExampleProfileList/members?action=delete
```

Sample request body:

```
{
  "queryAttribute" : "e",
  "ids" : ["recipient1@example.com", "recipient2@example.com"]
}
```

Sample Response in case of success:

RESPONSE NOTES:

- When the system successfully deletes records for a given ID (up to 10 records), then the ID is returned in the response.
- In a successful delete, the system deletes matching records for a given ID, including seed/proof records, up to the maximum of 10 records. If the system finds more than 10 records matching an ID, then the system will delete only the 10 most recent records. To ensure that all records associated with a given ID are deleted, you can re-run the request until all of the requested IDs return a `DELETEFAILED: No records found` message in the response.
- Other attributes in the response like `mapTemplateName` and `mergeRule` will have default values of null/false.

```
HTTPS response status code: 200 OK

{
  "recordData":
    {"fieldNames": ["EMAIL_ADDRESS_"],
     "records": [
       ["recipient1@example.com "],
       ["DELETEFAILED: No Records found for EMAIL_ADDRESS_ =
recipient2@example.com"]
     ],
     "mapTemplateName": null },

  "mergeRule":
    {"textValue": null,
     "htmlValue": null,
     "optinValue": null,
     "insertOnNoMatch": false,
     "updateOnMatch": null,
     "matchColumnName1": null,
     "matchColumnName2": null,
     "matchOperator": null,
     "optoutValue": null,
     "rejectRecordIfChannelEmpty": null,
     "defaultPermissionStatus": null,
     "matchColumnName3": null },
  "links": [
    { "rel": "self",
      "href":
"/rest/api/v1.3/lists/MyExampleProfileList/members?action=delete",
      "method": "POST" },
    { "rel": "retrieveMultipleListRecipients",
      "href":
"/rest/api/v1.3/lists/MyExampleProfileList/members?action=get",
      "method": "POST" },
    { "rel": "mergeListRecipients",
      "href": "/rest/api/v1.3/lists/MyExampleProfileList/members",
      "method": "POST" }
  ]
}
```

Troubleshooting error responses

Individual list member errors when the HTTPS response is `200 OK`

When the system has successfully received a request to delete multiple list members, it will return a successful HTTPS status code of `200 OK`. However, the batch delete might have mixed results. The response payload contains an array of messages about the success or failure for each individual member. Some examples of `DELETEFAILED` messages:

- When the system could not find any records corresponding to the ID, it returns an error message of `DELETEFAILED: No Records found`.
- When an invalid value is sent for the ID, the system returns a `DELETEFAILED` error for that ID. For example, if `test` is sent as one of the values in the `ids` array when the `queryAttribute` is `r`, then the system returns the following message for that ID: `DELETEFAILED: ERROR: The value test is not valid for an integer field\n\n`

Other error responses:

Record limit exceeded: The 200-record limit is exceeded (that is, more than 200 query ID values are sent). A `400 bad request` error is returned with the following error response body.

```
{
  "type": "",
  "title": "Record limit exceeded",
  "errorCode": "RECORD_LIMIT_EXCEEDED",
  "detail": "Record limit exceeded, maximum of 200 records are
allowed per each api call",
  "errorDetails": []
}
}
```

Invalid parameter: The `queryAttribute` value is not valid. A `400 bad request` error is returned with the following error response body:

```
{
  "type": "",
  "title": "Invalid request parameters",
  "errorCode": "INVALID_PARAMETER",
  "detail": "Query Attribute Must be either r, e, c or m",
  "errorDetails": []
}
```

API limit exceed: When the client application exceeds the throttling limit for this API, a `401 Unauthorized` error is returned with the following error response body:

```
{
  "type": "",
  "title": "",
  "errorCode": "API_LIMIT_EXCEEDED",
  "detail": "",
  "errorDetails": []
}
```

List not found: The system could not find the profile list specified in the request endpoint. A `404 not found error` is returned with the following error response body:

```
{
  "type": "",
  "title": "List not found",
  "errorCode": "LIST_NOT_FOUND",
  "detail": "MyOtherExampleProfileList List Not Found",
  "errorDetails": []
}
```

Managing Profile Extension Tables

For a given profile list table, its profile extension tables can be retrieved. In addition, 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

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

Service URL:

```
/rest/api/v1.3/lists/{listName}/listExtensions
```

Request Method:

```
GET
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request Body:

None

Sample Response in case of success:

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

```
[
  {
    "profileExtension": {
      "objectName": "Ax_PET_201905240806"
    },
    "fields": [
      {
        "fieldName": "RIID_",
        "fieldType": "INTEGER"
      },
      {
        "fieldName": "EMAIL_ADDRESS",
        "fieldType": "STR500"
      },
      {
        "fieldName": "CREATED_BY_LOAD_JOB_ID_",
        "fieldType": "INTEGER"
      },
      {
        "fieldName": "LAST_MOD_BY_LOAD_JOB_ID_",
        "fieldType": "INTEGER"
      },
      {
        "fieldName": "CREATED_DATE_",
        "fieldType": "TIMESTAMP"
      },
      {
        "fieldName": "MODIFIED_DATE_",
        "fieldType": "TIMESTAMP"
      },
      {
        "fieldName": "LAST_BULK_LOAD_ID_",
        "fieldType": "INTEGER"
      },
      {
        "fieldName": "AX_NEWPETFIELD1",
        "fieldType": "STR100"
      },
      {
        "fieldName": "AX_NEWPETFIELD2",
        "fieldType": "INTEGER"
      }
    ]
  },
  {
    "profileExtension": {
      "objectName": "Ax_PET_201905241137"
    },
    "fields": [
      {
        "fieldName": "RIID_",
        "fieldType": "INTEGER"
      },
      {
        "fieldName": "EMAIL_ADDRESS",
        "fieldType": "STR500"
      },
      {
        "fieldName": "CREATED_BY_LOAD_JOB_ID_",
        "fieldType": "INTEGER"
      },
      {
        "fieldName": "LAST_MOD_BY_LOAD_JOB_ID_",
        "fieldType": "INTEGER"
      },
      {
        "fieldName": "CREATED_DATE_",
        "fieldType": "TIMESTAMP"
      },
      {
        "fieldName": "MODIFIED_DATE_",
        "fieldType": "TIMESTAMP"
      },
      {
        "fieldName": "LAST_BULK_LOAD_ID_",
        "fieldType": "INTEGER"
      },
      {
        "fieldName": "AX_NEWPETFIELD1",
        "fieldType": "STR100"
      },
      {
        "fieldName": "AX_NEWPETFIELD2",
        "fieldType": "INTEGER"
      }
    ]
  }
]
```

Sample Response in case of failure:

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

Create a new profile extension table

You can create a new profile extension table for a given profile list by providing the schema of the profile extension table.

Service URL:

`/rest/api/v1.3/lists/{listName}/listExtensions`

Request Method:

`POST`

Request Body Properties:

- `objectName` - name of the Profile List.
- `folderName` - the name of the folder the Profile List is located.
- `fields` - the profile extension table fields. When creating a new profile extension table, the supported field types are:
  - `STR500`
  - `STR4000`
  - `INTEGER`
  - `NUMBER`
  - `TIMESTAMP`

Sample Request Body:

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

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Response if successful:

```
true
```

Sample Response if failure:

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

Merge or update members in a profile extension table

For an existing profile extension table, you can add new members or update data for existing members.

REQUEST NOTE: When creating a new profile extension table, the supported field types are:

- The service URL requires the names of *both* the Profile List and the Profile Extension Table.
- Up to 200 members can be processed in a single request.
- Up to two match columns can be used for merging records into a profile extension table. If only one match column is specified, the other match column can be set to null.
- For a given profile extension table, an array of record data that contain field names and their corresponding field values are specified. The `fieldNames` attribute must contain at least one of the merge key fields from the profile list (for example, `RIID_`, `EMAIL_ADDRESS_`, `CUSTOMER_ID_`), otherwise the request will result in a `MERGEFAILED` error.

- `matchColumnName1` and `matchColumnName2` values must not contain the usual trailing underscore for the enumerated values. For example, `RIID` works but `RIID_` results in an `INVALID_PARAMETER` error ("Match Column is null"). `matchColumnName` values can be as follows: `RIID`, `CUSTOMER_ID`, `EMAIL_ADDRESS`, `MOBILE_NUMBER`, `EMAIL_MD5_HASH`, or `EMAIL_SHA256_HASH`
- Limitation for match columns: New records will not be inserted for `insertOnNoMatch` if a `matchColumnName` 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.

Please see the definition of merge rule parameters provided in Definitions of Rule Parameters for Merging Members into a Profile List.

Service URL:

`/rest/api/v1.3/lists/{listName}/listExtensions/{petName}/members`

Request Method:

`POST`

Request Header:

```
Authorization=<AUTH_TOKEN>
Content-Type=application/json
```

Request Body:

```
{
                "recordData" : {
                 "fieldNames" : ["riid_", "email_address_", "salary"],
                 "records" : [
                         ["1761408", "foo@bar.com", "10000"],
                         ["98798", "baz@foo.com", "298909"],
                         ["xyz", "bar@baz.com", "wedrfwe"],
                         ["12312", "bar@foo.com", "23423","23423"],
                         ["1761409","foo@baz.com", "239482734"]
                      ],
                       "mapTemplateName" : null
                },
                "insertOnNoMatch" : true,
                "updateOnMatch" : "REPLACE_ALL",
                "matchColumnName1" : "RIID",
                "matchColumnName2" : "EMAIL_ADDRESS"
}
```

Sample Response in case of success:

- Irrespective of what field names were used to perform the merge, the response will always contain only `RIID_` in the `fieldNames` attribute and the corresponding RIID_ values for the records in the `records` attribute.
- In case merge failed for a record, the `RIID_` of the record is not present in the response. Instead, an error message starting with `MERGEFAILED:` is returned. Client developers can look for the string `MERGEFAILED:` in the response for a particular row to determine whether that recipient was merged successfully or not.
- **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`, `matchColumnName1`, and `matchColumnName2` will mirror the valid values specified in the request payload or default to null/false in case of invalid values.

```
{
   "recordData":    {
      "fieldNames": ["RIID_"],
      "records":       [
         ["1761408"],
         ["MERGEFAILED: Record 1 = RECORD DOES NOT MATCH ANY
CONTACTS IN THE LIST DemoNewsLetterList\r\n"],
         ["MERGEFAILED: Record 2 = ERROR: The value xyz is not
valid for an integer field\n\n\r\n"],
         ["MERGEFAILED: Record 3 = Field Names length, doesn't
match with Field Values length\r\n"],
         ["1761409"]
      ],
      "mapTemplateName": null
   },
   "insertOnNoMatch": true,
   "updateOnMatch": "REPLACE_ALL",
   "matchColumnName1" : "RIID",
   "matchColumnName2" : "EMAIL_ADDRESS"
   "links":    [
      {
         "rel": "self",
         "href":
"/rest/api/v1.3/lists/DemoNewsLetterList/listExtensions/DemoNewsLet
terPet/members",
         "method": "POST"
      },
      {
         "rel": "retrieveProfileExtensionRecipientsRIID",
         "href":
"/rest/api/v1.3/lists/DemoNewsLetterList/listExtensions/DemoNewsLet
terPet/members/<riid>",
         "method": "GET"
      },
      {
         "rel": "deleteProfileExtensionRecipientsRIID",
         "href":
"/rest/api/v1.3/lists/DemoNewsLetterList/listExtensions/DemoNewsLet
terPet/members/<riid>",
         "method": "DELETE"
      }
   ]
}
```

Sample Response in case of failure:

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

Retrieve a member of a profile extension table based on RIID

You can retrieve existing members of a profile extension table one at a time by specifying the member's Responsys ID (RIID).

REQUEST NOTES:

- The service URL requires the names of *both* the Profile List and the Profile Extension Table.
- 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/v1.3/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'
```

Request Body:

None

Sample Requests:

To retrieve all fields:

```
/rest/api/v1.3/lists/DemoNewsLetterList/listExtensions/DemoNewsLetter
Pet/members/1761408?fs=all
```

To receive the fields Salary and RIID_:

```
/rest/api/v1.3/lists/DemoNewsLetterList/listExtensions/DemoNewsLetter
Pet/members/1761408?fs=salary,riid_
```

Sample Response in case of success:

RESPONSE NOTE: Other attributes in the response, such as `mapTemplateName`, `insertOnNoMatch`, `updateOnMatch`, `matchColumnName1`, and `matchColumnName2` will have default values (`null`/`false`).

```
{
  "recordData": {
    "fieldNames": [
      "RIID_",
      "SALARY"
    ],
    "records": [
      [
        "1761409",
        "239482734"
      ]
    ],
    "mapTemplateName": null
  },
  "insertOnNoMatch": false,
  "updateOnMatch": null,
  "matchColumnName1": null,
  "matchColumnName2": null,
  "links": [
    {
      "rel": "self",
      "href":
"/rest/api/v1.3/lists/DemoNewsLetterList/listExtensions/DemoNewsLet
terPet/members?qa=e&amp;fs=riid_,salary&amp;id=responsysblr@gmail.c
om",
      "method": "GET"
    },
    {
      "rel": "mergeProfileExtensionRecipients",
      "href":
"/rest/api/v1.3/lists/DemoNewsLetterList/listExtensions/DemoNewsLet
terPet/members",
      "method": "POST"
    },
    {
      "rel": "deleteProfileExtensionRecipientsRIID",
      "href":
"/rest/api/v1.3/lists/DemoNewsLetterList/listExtensions/DemoNewsLet
terPet/members/&lt;riid&gt;",
      "method": "DELETE"
    }
  ]
}
```

Sample Responses in case of failure:

When the system cannot find a record matching the ID that you sent:

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

If the `fs` parameter is omitted:

```
{
  "type": "",
  "title": "Invalid request parameters",
  "errorCode": "INVALID_PARAMETER",
  "detail": "FieldList is null OR empty",
  "errorDetails": []
}
```

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

You can retrieve existing members of a profile extension table one at a time by specifying a unique identifier other than the member’s Responsys ID (RIID) through the query attribute of the interface.

REQUEST NOTES:

- The service URL requires the names of *both* the Profile List and the Profile Extension Table.
- 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/v1.3/lists/{listName}/listExtensions/
{petName}/members
```

Request Method:

```
GET
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

Request parameters:

- `qa` - Query Attribute. Can be one of the following values:
  - ‘r’ - RIID
  - ‘e’ - EMAIL_ADDRESS

    - ‘c’ - CUSTOMER_ID
    - ‘m’ - MOBILE_NUMBER
- `id` - ID corresponding to the query attribute.
- `fs` - Comma separated field list or `all`.

Request Body:

None

Sample Response in case of success:

RESPONSE NOTE: Other attributes in the response, such as `mapTemplateName`, `insertOnNoMatch`, `updateOnMatch`, `matchColumnName1`, and `matchColumnName2`will have default values (`null`/`false`).

```
{
  "recordData": {
    "fieldNames": [
      "RIID_",
      "SALARY"
    ],
    "records": [
      [
        "1761409",
        "239482734"
      ]
    ],
    "mapTemplateName": null
  },
  "insertOnNoMatch": false,
  "updateOnMatch": null,
  "matchColumnName1": null,
  "matchColumnName2": null,
  "links": [
    {
      "rel": "self",
      "href":
"/rest/api/v1.3/lists/DemoNewsLetterList/listExtensions/DemoNewsLet
terPet/members?qa=e&amp;fs=riid_,salary&amp;id=responsysblr@gmail.c
om",
      "method": "GET"
    },
    {
      "rel": "mergeProfileExtensionRecipients",
      "href":
"/rest/api/v1.3/lists/DemoNewsLetterList/listExtensions/DemoNewsLet
terPet/members",
      "method": "POST"
    },
    {
      "rel": "deleteProfileExtensionRecipientsRIID",
      "href":
"/rest/api/v1.3/lists/DemoNewsLetterList/listExtensions/DemoNewsLet
terPet/members/&lt;riid&gt;",
      "method": "DELETE"
    }
  ]
}
```

Sample Responses in case of failure:

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

Delete a member of a profile extension table based on RIID

You can delete existing members of a profile extension table one at a time by specifying the Responsys ID (RIID).

Service URL:

`/rest/api/v1.3/lists/{listName}/listExtensions/{petName}/members/{riid}`

Request Method:

`DELETE`

Request Header:

`Authorization=<AUTH_TOKEN>`

Request Body:

None

Sample Response in case of success:

REQUEST NOTES:

- The service URL requires the names of *both* the Profile List and the Profile Extension Table.
- 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 for some reason, 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.
- Other attributes in the response (`mapTemplateName`, `insertOnNoMatch`, `updateOnMatch`, `matchColumnName1`, `matchColumnName2`) will have default values (`null`/`false`).

```
{
  "recordData": {
    "fieldNames": [
      "RIID_"
    ],
    "records": [
      [
        "DELETEFAILED: NO records found for id\n"
      ]
    ],
    "mapTemplateName": null
  },
  "insertOnNoMatch": false,
  "updateOnMatch": null,
  "matchColumnName1": null,
  "matchColumnName2": null,
  "links": [
    {
      "rel": "self",
      "href":
"/rest/api/v1.3/lists/DemoNewsLetterList/listExtensions/DemoNewsLet
terPet/members/1761409",
      "method": "DELETE"
    },
    {
      "rel": "mergeProfileExtensionRecipients",
      "href":
"/rest/api/v1.3/lists/DemoNewsLetterList/listExtensions/DemoNewsLet
terPet/members",
      "method": "POST"
    },
    {
      "rel": "retrieveProfileExtensionRecipientsRIID",
      "href":
"/rest/api/v1.3/lists/DemoNewsLetterList/listExtensions/DemoNewsLet
terPet/members/&lt;riid&gt;",
      "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,
  "matchColumnName1": null,
  "matchColumnName2": null,
  "links": [
    {
      "rel": "self",
      "href":
"/rest/api/v1.3/lists/DemoNewsLetterList/listExtensions/DemoNewsLet
terPet/members/1761409",
      "method": "DELETE"
    },
    {
      "rel": "mergeProfileExtensionRecipients",
      "href":
"/rest/api/v1.3/lists/DemoNewsLetterList/listExtensions/DemoNewsLet
terPet/members",
      "method": "POST"
    },
    {
      "rel": "retrieveProfileExtensionRecipientsRIID",
      "href":
"/rest/api/v1.3/lists/DemoNewsLetterList/listExtensions/DemoNewsLet
terPet/members/&lt;riid&gt;",
      "method": "GET"
    }
  ]
}
```

Sample Responses in case of failure:

When the system cannot find a record matching the ID that you sent:

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

If the `fs` parameter is omitted:

```
{
  "type": "",
  "title": "Invalid request parameters",
  "errorCode": "INVALID_PARAMETER",
  "detail": "FieldList is null OR empty",
  "errorDetails": []
}
```

Delete multiple profile extension table records in a single request

Your client application can delete multiple profile extension table (PET) records in a single batch request by using the query attribute `action=delete` and by passing a list of IDs.

REQUEST NOTES:

- For this request to work, your endpoint must include the query attribute `action=delete` (as shown in the Service URL below).
- Note that the request method for this API is `POST`, because you pass the query attribute and the IDs to delete in the request body.
- You can send up to 200 IDs in the request body.

Service URL:

`/rest/api/v1.3/lists/{listName}/listExtensions/{petName}/members`**`?action=delete`**

Request Method:

`POST`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Request Body:

Request attributes:

`queryAttribute` - Specifies the query attribute to use for locating the records that you want to delete. All query attributes correspond to the PET's parent **profile list** fields. The system uses the query attribute to look up the profile list record, and then deletes the corresponding PET record. It can be one of the following values:

- `r` - RIID of the parent profile list record
- `e` - Email address (from the parent profile list)
- `c` - Customer ID (from the parent profile list)
- `m` - Mobile number (from the parent profile list)

`ids` - Array containing the identifier values corresponding to the query attribute. For example, if you specify `c`, the system expects the `ids` array to contain the customer ID values for the records you want to delete. You can pass up to 200 IDs in the request, and each ID must be 500 characters or fewer.

Sample request endpoint and body:

When Responsys receives the following sample endpoint and request, the system will attempt to delete records from the profile extension table MyExamplePET that correspond to records in the profile list **MyExampleProfileList**.

1. Because the query attribute is **r** and the IDs are `63036487` and `63027514`, the system searches **MyExampleProfileList** for those RIIDs.
2. Next, the system locates the **MyExamplePET** records associated with the profile list records that it located.
3. Finally, the system deletes the PET records that it found.

Sample request endpoint:

```
POST
/rest/api/v1.3/lists/MyExampleProfileList/listExtensions/MyExampleP
ET/members?action=delete
```

Sample request body:

```
{
  "queryAttribute" : "r",
  "ids" : ["63036487", "63027514"]
}
```

Sample Response in case of success:

RESPONSE NOTES:

- When the system successfully deletes records for a given ID (up to 10 records), then the ID is returned in the response.
- In a successful delete, the system deletes matching records for a given ID, including seed/proof records, up to the maximum of 10 records. If the system finds more than 10 records matching an ID, then the system will delete only the 10 most recent records. To ensure that all records associated with a given ID are deleted, you can re-run the request until all of the requested IDs return a `DELETEFAILED: No records found` message in the response.
- Other attributes in the response like `mapTemplateName`, `insertOnNoMatch`, `updateOnMatch`, `matchColumnName1`, and `matchColumnName2` will have default values of null/false.

The following response shows that the delete succeeded for records with RIID values of `63036487` and `63027514`.

```
HTTPS response status code: 200 OK

{
  "recordData":
    {"fieldNames": ["RIID_"],
     "records": [
       ["63036487"],
       ["63027514"]
     ],
     "mapTemplateName": null },

  "mergeRule":
    {"insertOnNoMatch": null,
     "updateOnMatch": null,
     "matchColumnName1": null,
     "matchColumnName2": null },
  "links": [
    { "rel": "self",
      "href":
"/rest/api/v1.3/lists/MyExampleProfileList/listExtensions/MyExample
PET/members?action=delete",
      "method": "POST" },
    { "rel": "mergeProfileExtensionRecipients",
      "href":
"/rest/api/v1.3/lists/MyExampleProfileList/listExtensions/MyExample
PET/members",
      "method": "POST" },
    { "rel": "retrieveProfileExtensionRecipients",
      "href":
"/rest/api/v1.3/lists/MyExampleProfileList/listExtensions/MyExample
PET/members",
      "method": "GET" }
  ]
}
```

Troubleshooting error responses

Individual list member errors when the HTTPS response is `200 OK`

When the system has successfully received a request to delete multiple list members, it will return a successful HTTPS status code of `200 OK`. However, the batch delete might have mixed results. The response payload contains an array of messages about the success or failure for each individual member. Some examples of `DELETEFAILED` messages:

- When the system could not find any records corresponding to the ID, it returns an error message of `DELETEFAILED: No Records found for id\n`.

- When an invalid value is sent for the ID, or the value is longer than the maximum field length, the system returns a `DELETEFAILED` error for that ID. For example, if `test` is sent as one of the values in the `ids` array when the `queryAttribute` is `r`, then the system returns the following message for that ID: `DELETEFAILED: ERROR: The value test is not valid for an integer field\n\n`

Other error responses:

Record limit exceeded: The 200-record limit is exceeded (that is, more than 200 query ID values are sent). A `400 bad request` error is returned with the following error response body.

```
{
  "type": "",
  "title": "Record limit exceeded",
  "errorCode": "RECORD_LIMIT_EXCEEDED",
  "detail": "Record limit exceeded, maximum of 200 records are
allowed per each api call",
  "errorDetails": []
}
}
```

Invalid parameter: The `queryAttribute` value is not valid. A `400 bad request` error is returned with the following error response body:

```
{
  "type": "",
  "title": "Invalid request parameters",
  "errorCode": "INVALID_PARAMETER",
  "detail": "Query Attribute Must be either r, e, c or m",
  "errorDetails": []
}
```

API limit exceed: When the client application exceeds the throttling limit for this API, a `401 Unauthorized` error is returned with the following error response body:

```
{
  "type": "",
  "title": "",
  "errorCode": "API_LIMIT_EXCEEDED",
  "detail": "",
  "errorDetails": []
}
```

PET not found: The system could not find the PET specified in the request endpoint. A `404 not found error` is returned with the following error response body:

```
{
  "type": "",
  "title": "Profile list extension not found",
  "errorCode": "PROFILE_EXTENSION_NOT_FOUND",
  "detail": "MyOtherExamplePET Profile Extension Not Found",
  "errorDetails": []
}
```

List not found: The system could not find the profile list specified in the request endpoint. A `404 not found error` is returned with the following error response body:

```
{
  "type": "",
  "title": "List not found",
  "errorCode": "LIST_NOT_FOUND",
  "detail": "MyOtherExampleProfileList List Not Found",
  "errorDetails": []
}
```

Add or update RIID-to-Audience Scope Code mapping data for profile list members

Important: This API only applies if Targeting by Organization is enabled for your account.

This API allows you to update the system mapping table that maps a Profile List member's RIID to the Organization Scope appropriate for that member. This ensures that Responsys only sends messages to those recipients belonging to a campaign's target audience of selected organization units.

REQUEST NOTES:

- The service URL only requires the names of the Profile List. The mapping table that stores the information is an internal system table that is not accessible through the application UI.
- Up to 200 members can be processed in a single request.
- Up to two match columns can be used for merging records into the table. If only one match column is specified, the other match column can be set to an empty value (for example, "`matchColumnName2`" : "").
- Mandatory `fieldNames` attributes:
  - `AUDIENCE_SCOPE_CODE`: This is the audience scope code, as defined in the organization table. It is a code that defines the org to which the customer belongs. You can view the organization tree when logged in to Responsys by selecting Account | Organization Management.
  - `REC_STATUS_`: This is the status of the record. Valid values are "`A`" (Active) or "`D`" (Deleted).
    - **To add a mapping record**: Specify "`A`" for REC_STATUS_. The system adds the mapping to the table. Note that each RIID (customer) can have more than one audience scope coding.
    - **To mark a record as deleted**: Specify "`D`" for REC_STATUS_. The system will treat this record as deleted.

    - Must include a system column name to be used for the merge, and it should also be used for `matchColumnName1`.

- `matchColumnName1` and `matchColumnName2` values must not contain the usual trailing underscore for the enumerated values. For example, `RIID` works but `RIID_` results in an `INVALID_PARAMETER` error (“Match Column is null”). `matchColumnName` values can be as follows: `RIID`, `CUSTOMER_ID`, `EMAIL_ADDRESS`, `MOBILE_NUMBER`, `EMAIL_MD5_HASH`, or `EMAIL_SHA256_HASH`
- Limitation for match columns: New records will not be inserted for `insertOnNoMatch` if a `matchColumnName` 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.

Please see the definition of merge rule parameters provided in Definitions of Rule Parameters for Merging Members into a Profile List.

Service URL:

`/rest/api/v1.3/lists/{listName}/orgListExtensions/members`

Request Method:

`PUT`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Request Body Example:

In the example below, both records have an `AUDIENCE_SCOPE_CODE` of "`IN`". The first example is a valid "Active" (add) entry, but the second record illustrates an incorrect `REC_STATUS_`, "F", which will result in the `INVALID_PARAMETER` error shown in the Sample Response Body.

```
{
  "recordData" : {
    "fieldNames" : ["AUDIENCE_SCOPE_CODE", "REC_STATUS_", "EMAIL_
ADDRESS_"],
    "records" : [
      ["IN", "A", "4as0dsa@yahoo.com"],
      ["IN", "F", "abcd@gmail.com"]
    ],
    "mapTemplateName" : null
  },
  "insertOnNoMatch" : true,
  "updateOnMatch" : "REPLACE_ALL",
  "matchColumnName1" : "EMAIL_ADDRESS"
  "matchColumnName2" : ""
}
```

Response:

For each record, if the Audience Scope Code is valid, then system updates the table. The system updates the table with the Profile List member's RIID, Audience Scope Code, and status details (`REC_STATUS_`).

- Irrespective of what field names were used to perform the merge, the response will always contain only `RIID_` in the `fieldNames` attribute and the corresponding RIID_ values for the records in the `records` attribute.

- In case merge failed for a record, the `RIID_` of the record is not present in the response. Instead, an error message starting with `MERGEFAILED:` is returned. Client developers can look for the string `MERGEFAILED:` in the response for a particular row to determine whether that recipient was merged successfully or not.

  Reasons for MERGEFAILED include an invalid REC_STATUS_ value, Audience Scope Code provided does not exist, and Audience Scope Code provided does not exist AND the member is not found in the Profile List.

- **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.

Sample response – successful merge for the first record, failed merge for the second record:

In this example, we sent two records (per the previous request example) to merge to the mapping table. The request was sent to the endpoint using the Profile List "contacts_list" as the `{list_name}` value. The list name is used in the endpoints provided in the `links` array in the response.

```
{
  "recordData":
    { "fieldNames": [ "RIID_" ],
      "records": [
        [ "8381" ],
        [ "MERGEFAILED: Invalid REC_STATUS_ provided - should be
'A' or 'D' " ]
  ],
    "mapTemplateName": null },

  "insertOnNoMatch": true,
  "updateOnMatch": "REPLACE_ALL",
  "matchColumn": "EMAIL_ADDRESS",

  "links": [
    { "rel": "self", "href": "/rest/api/v1.3/lists/contacts_
list/orgListExtensions/members", "method": "PUT" },
    { "rel": "retrieveProfileExtensionRecipientsRIID", "href":
"/rest/api/v1.3/lists/ contacts_
list/listExtensions//members/<riid>", "method": "GET"}
  ]
}
```

ERROR NOTES

- The sample on the next page shows the format for the error response body.
- Common errors returned when the request fails:

Error Code	Detail
RECORD_LIMIT_EXCEEDED	Record limit exceeded, maximum of 200 records are allowed per each API call
INVALID_PARAMETER	Match column name is not in `fieldNames`
INVALID_PARAMETER	Match column name is empty or is not a system column
INVALID_PARAMETER	`fieldNames` should have `REC_STATUS_` and `AUDIENCE_SCOPE`
PROFILE_EXTENSION_ NOT_FOUND	No Organization Profile Extensions Found For List <`list_name`>.

Sample Response in case of failure - request body attempted to send more than 200 recipients:

```
{
   "type": "",
   "title": "Record limit exceeded",
   "errorCode": "RECORD_LIMIT_EXCEEDED",
   "detail": "Record limit exceeded, maximum of 200 records are
allowed per each api call",
   "errorDetails": []
}
```

Managing Supplemental Tables

You can retrieve all supplemental data for your account, and you can create new supplemental tables. For an existing supplemental table, its members can be added, updated, retrieved, or deleted.

Retrieve all supplemental tables

Use this interface to get all supplemental table data for your account.

Service URL:

`/rest/api/v1.3/suppData`

Request Method:

`GET`

Query Parameters

- `offset`: starts at 0 and indicates the record number for the response result set (defaults to 0)
- `limit`: number of Supplemental Table records to return in the response (defaults to 200 and cannot exceed 200)

Request Header:

```
Authorization=<AUTH_TOKEN>

Content-Type=application/json
```

Sample Request Body:

`Not applicable`

Sample Response in case of success:

RESPONSE NOTES:

- The system returns an array of `suppData` objects. These objects contain the data for the supplemental table:
  - `objectName`: Name of the supplemental table
  - `folderName`: Name of the folder containing the supplemental table
  - `fields`: An array containing the supplemental table’s field properties. Key-value pairs are `fieldname` and `fieldType` (data type of the field).

- If your Responsys account does not have supplemental tables defined, then the response is an empty array.
- For accounts enabled for Organizational Access Control, the organization filter is applied before sending the response.

```
[
    {
        "suppData": {
            "objectName": "MY_SUPPLEMENTAL",
            "folderName": "myExample"
        },
        "fields": [
            {
                "fieldName": "EMAIL_ADDRESS",
                "fieldType": "STR500"
            },
            {
                "fieldName": "AGE",
                "fieldType": "NUMBER"
            },
            {
                "fieldName": "ARRIVAL",
                "fieldType": "TIMESTAMP"
            },
            {
                "fieldName": "DEPARTURE",
                "fieldType": "TIMESTAMP"
            },
            {
                "fieldName": "CITY",
                "fieldType": "STR25"
            },
            {
                "fieldName": "STATE",
                "fieldType": "STR25"
            },
            {
                "fieldName": "EVENT",
                "fieldType": "STR25"
            },
            {
                "fieldName": "EVENT_ID",
                "fieldType": "INTEGER"
            },
            {
                "fieldName": "CREATED_DATE_",
                "fieldType": "TIMESTAMP"
            },
            {
                "fieldName": "MODIFIED_DATE_",
                "fieldType": "TIMESTAMP"
            }
        ]
    },
    {
        "suppData": {
            "objectName": "jmp supp table",
            "folderName": "JMP test"
        },
        "fields": [
            {
                "fieldName": "ANIMAL",
                "fieldType": "STR25"
            },
            {
                "fieldName": "VEGETABLE",
                "fieldType": "STR100"
            },
            {
                "fieldName": "MINERAL",
                "fieldType": "STR25"
            },
            {
                "fieldName": "NUMERAL",
                "fieldType": "INTEGER"
            },
            {
                "fieldName": "CREATED_DATE_",
                "fieldType": "TIMESTAMP"
            },
            {
                "fieldName": "MODIFIED_DATE_",
                "fieldType": "TIMESTAMP"
            }
        ]
    },
    {
        "suppData": {
            "objectName": "example_supp",
            "folderName": "ExampleFolder"
        },
        "fields": [
            {
                "fieldName": "ORDER_ID",
                "fieldType": "NUMBER"
            },
            {
                "fieldName": "PRICE",
                "fieldType": "NUMBER"
            },
            {
                "fieldName": "RIID_",
                "fieldType": "NUMBER"
            },
            {
                "fieldName": "CREATED_DATE_",
                "fieldType": "TIMESTAMP"
            },
            {
                "fieldName": "MODIFIED_DATE_",
                "fieldType": "TIMESTAMP"
            }
        ]
    }
]
```

Create a new supplemental table

Use this interface to create a new supplemental table by providing its schema.

Service URL:

`/rest/api/v1.3/folders/{folderName}/suppData`

Request Method:

`POST`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Sample Request Body:

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

Response if successful:

```
true
```

Sample Response if failed:

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

Get primary keys or all field names of a supplemental table

For an existing supplemental table, you can get only the primary key field names or all of the field names.

Service URL:

`/rest/api/v1.3/folder/{folderName}/suppData/{suppTableName}`

Request Method:

`GET`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Sample Request Body:

```
        Not applicable
```

Response Notes

- The system returns an array of `fields`. Depending on the `ft` query parameter, the response will send either all of the fields or only the primary key fields. For each field returned, the key-value pairs are `fieldName` and `fieldType` (data type of the field).
- The system returns an array of `suppData` objects. These objects contain the data for the supplemental table:
  - `objectName`: Name of the supplemental table
  - `folderName`: Name of the folder containing the supplemental table

Sample response in case of success:

To get only the primary key fields for a supplemental table named **MyExampleSuppTable** in the folder **MyExampleFolder**, a client application sent a `GET` request as follows:

```
GET
/rest/api/v1.3/folder/MyExampleFolder/suppData/MyExampleSuppTable?f
t=PK
```

The system sent the following response, where the fields array contains the primary key field names and their types. In this example, the supplemental table had one primary key field, `MyPKfield1`, of type `STR500`. The `suppData` object echoes back the folder name and supplemental table name sent in the request.

```
{
  "fields": [
    { "fieldName": "MyPKfield1", "fieldType": "STR500" }
  ],
  "suppData":
    { "folderName": "MyExampleFolder", "objectName":
"MyExampleSuppTable" },
  "links": [
    { "rel": "self",
      "href":
"/rest/api/v1.3/folder/MyExampleFolder/suppData/MyExampleSuppTabl
e",
      "method": "GET" },
    { "rel": "mergeTableMembers",
      "href":
"/rest/api/v1.3/folders/MyExampleFolder/suppData/MyExampleSuppTable
/members",
      "method": "POST" },
    { "rel": "retrieveTableMembers",
      "href":
"/rest/api/v1.3/folders/MyExampleFolder/suppData/MyExampleSuppTable
/members",
      "method": "GET" },
    { "rel": "deleteTableMembers",
      "href":
"/rest/api/v1.3/folders/MyExampleFolder/suppData/MyExampleSuppTable
/members",
      "method": "DELETE" }
  ]
}
```

Troubleshooting error responses:

The system returns error responses in the following situations.

The folder does not exist for the Responsys account: A `404 not found` error is returned with the following error response body (where the folder name sent is returned in the square brackets):

```
{
  "type": "",
  "title": "Folder not found",
  "errorCode": "FOLDER_NOT_FOUND",
  "detail": "Folder name [MyExampleFolder] does not exist",
  "errorDetails": []
}
```

The table does not exist for the Responsys account: A `404 not found` error is returned with the following error response body (where the table name sent is returned in the square brackets):

```
{
  "type": "",
  "title": "Resource not found",
  "errorCode": "RESOURCE_NOT_FOUND",
  "detail": "Supplemental name [MyExampleSuppTable] does not
exist",
  "errorDetails": []
}
```

An invalid query parameter is sent: That is, `ft` is set to a value other than `pk`). A `400 bad request` error is returned with the following error response body:

```
{
  "type": "",
  "title": "Invalid request parameters",
  "errorCode": "INVALID_PARAMETER",
  "detail": "Not a valid value for Request parameter ft. Allowed
value is PK",
  "errorDetails": []
}
```

Organization filter error: For accounts enabled for Organizational Access Control, the system applies the organization filter. So when a user belonging to an organization tries to invoke the API on a supplemental table belonging to another organization, the system returns the following error (where the table name sent is returned in the square brackets):

```
{
  "type": "",
  "title": "Resource not found",
  "errorCode": "RESOURCE_NOT_FOUND",
  "detail": "Supplemental name [ExampleNotMyOrgTable] does not
exist",
  "errorDetails": []
}
```

Merge supplemental table records using primary key

For an existing supplemental table, you can add new members or update data for existing members.

REQUEST NOTES:

- For a given supplemental table in a specific folder, you must specify an array of record data that contains field names and their corresponding field values along with ALL primary keys to identify the desired record.
- A maximum of 200 records can be merged in one request.
- All the Primary Key Fields of the table must be specified in the `fieldnames` attribute along with their corresponding values in the `records` attribute.
- Field names specified in the `fieldNames` attribute are case sensitive. They must match their case as defined in the supplemental table.

Service URL:

```
/rest/api/v1.3/folders/{folderName}/suppData/
{tableName}/members
```

Request Method:

```
POST
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

```
Content-Type=application/json
```

Sample Request Body:

```
    {
              "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:

RESPONSE NOTES:

- 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.
- **The order of records in the response matches the order of records specified in the request payload.**
- Other attributes in the response, such as `mapTemplateName`, `insertOnNoMatch`, `updateOnMatch`, and `matchColumn`, will mirror the valid values specified in the request payload **or** default to null/false in case of invalid values.

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

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

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

```
{
   "recordData":    {
      "fieldNames":       [
         "PK1",
         "PK2",
         "F1",
         "F2"
      ],
      "records":       [
                  [
            "MERGEFAILED: Record 0 = NOT UPDATED PER MERGE RULE.
MATCH FIELD CANNOT BE EMPTY\r\n",
            "",
            "",
            ""
         ],
                  [
            "MERGEFAILED: Record 1 = NOT UPDATED PER MERGE RULE.
MATCH FIELD CANNOT BE EMPTY\r\n",
            "",
            "",
            ""
         ],
                  [
            "MERGEFAILED: Record 2 = NOT UPDATED PER MERGE RULE.
MATCH FIELD CANNOT BE EMPTY\r\n",
            "",
            "",
            ""
         ]
      ],
      "mapTemplateName": null
   },
   "insertOnNoMatch": true,
   "updateOnMatch": "REPLACE_ALL",
   "links": [   {
      "rel": "self",
      "href":
"/rest/api/v1.3/folders/DemoNewsLetter/suppData/CompositePKSuppTabl
e/members",
      "method": "POST"
   }]
}
```

Sample Response in case of failure:

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

Merge supplemental table records without primary key

For an existing supplemental table, you can add new members or update data for existing members.

REQUEST NOTES:

- For a given supplemental table in a specific folder, you must specify an array of record data that contains field names and their corresponding field values using a `matchColumnNames` attribute in the payload.
- A maximum of 200 records can be merged in one request.
- Use the `matchColumnNames` attribute in the payload to identify the column to use to match to a record in the supplemental table without a primary key.

- This interface enforces that `insertOnNoMatch` is true and `updateOnNoMatch` is “`REPLACE_ALL`” in the request payload.
- Field names specified in the `fieldNames` attribute are case sensitive. They must match their case as defined in the supplemental table.

Service URL:

`/rest/api/v1.3/folders/{folderName}/suppData/{tableName}/members`

Request Method:

`POST`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Sample Request Body:

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

Sample Response in case of success:

RESPONSE NOTES:

- IWhen 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.
- **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/v1.3/folders/DemoNewsLetter/suppData/CompositePKSuppTabl
e/members",
      "method": "POST"
   }]
}
```

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

Sample Payload:

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

Response:

```
  {
   "recordData":    {
      "fieldNames":       [
         "PK1",
         "PK2",
         "F1",
         "F2",
         "PK3"
      ],
      "records":       [
                  [
            "MERGEFAILED: Unable to identify record to Merge from
the Match Columns and FieldNames Specified.",
            "",
            "",
            "",
            ""
         ],
                  [
            "MERGEFAILED: Unable to identify record to Merge from
the Match Columns and FieldNames Specified.",
            "",
            "",
            "",
            ""
         ],
                 [
            "MERGEFAILED: Unable to identify record to Merge from
the Match Columns and FieldNames Specified.",
            "",
            "",
            "",
            ""
         ]
      ],
      "mapTemplateName": null
   },
   "insertOnNoMatch": true,
   "updateOnMatch": "REPLACE_ALL",
   "matchColumnNames": ["F1"],
   "links": [   {
      "rel": "self",
      "href":
"/rest/api/v1.3/folders/DemoNewsLetter/suppData/CompositePKSuppTabl
e/members",
      "method": "POST"
   }]
}
```

Sample Response in case insertOnMatch or updateOnNoMatch are invalid:

Sample Payload:

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

Response:

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

Sample Response in case matchColumnNames attribute is not specified:

Sample Payload:

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

Response:

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

Retrieve supplemental table records with primary key

Use this interface to retrieve Supplemental Table Records by specifying the primary key values using the request parameters.

REQUEST NOTES:

- 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/v1.3/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.

Request Body:

None

Sample Request:

```
/rest/api/v1.3/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:

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

```
{
    "recordData":{
        "fieldNames":[
            "PK1",
            "PK2",
            "PK3",
            "F1",
            "F2"
        ],
        "records":[
        [
                "1",
                "1",
                "1",
                "onerec",
                "onecol"
         ]
        ],
        "mapTemplateName":null
    },
    "insertOnNoMatch":false,
    "updateOnMatch":null,
    "links":[
        {
            "rel":"self",

"href":"/rest/api/v1.3/folders/DemoNewsLetter/suppData/CompositePKS
uppTable/members?qa=PK1&qa=PK2&qa=PK3&fs=PK1,PK2,PK3,F1,F2&id=1&id=
1&id=1",
            "method":"GET"
        },
        {
            "rel":"mergeTableMembers",
"href":"/rest/api/v1.3/folders/DemoNewsLetter/suppData/CompositePKS
uppTable/members",
            "method":"POST"
        }
    ]
}
```

Sample Response in case of failure:

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

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

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

Delete supplemental table records

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

Service URL:

`/rest/api/v1.3/folders/{folderName}/suppData/{tableName}/members`

Request Method:

`DELETE`

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. Do not send other field names
- `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.

Request Body:

None

Sample Request:

```
rest/api/v1.3/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:

RESPONSE NOTES:

- 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 is returned that states, "`No Records found for the Id`".
- 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.

- Other attributes in the response, such as `mapTemplateName`, `insertOnNoMatch`, `updateOnMatch`, and `matchColumn`, will have default values (null/false).

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

"href":"/rest/api/v1.3/folders/DemoNewsLetter/suppData/CompositePKS
uppTable/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/v1.3/folders/DemoNewsLetter/suppData/CompositePKS
uppTable/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 not ALL Primary Key Columns are 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": []
}
```

Campaigns

You can use the **Get All Campaigns** API (`GET /rest/api/v1.3/campaigns?type={campaign_type}`) to get a list of all EMD email, Push, Message Center, SMS, or MMS campaigns from Responsys. This helps you obtain the correct campaign names to use in other API requests. The following sections describe how to fetch campaigns for each supported campaign type.

Get a campaign

Use this interface to get an existing EMD Email campaign object. The response returns the campaign object, which includes the campaign ID and the campaign's other properties. The links array contains the campaign object's related API operations, specific to the campaign name where applicable.

Service URL:

```
/rest/api/v1.3/campaigns/{campaignName}
```

Required Path Parameters:

`campaignName` - Name of the campaign to be fetched.

Request Method:

```
GET
```

Request Header:

```
Authorization=<AUTH_TOKEN>

Content-Type=application/json
```

Request Body - Required Properties:

```
Not applicable
```

Sample Request URL:

```
/rest/api/v1.3/campaigns/example_campaign
```

Sample Response Body

```
{
  "id": 1040,
  "name": "example_campaign",
  "folderName": "Folder1",
  "type": "EMAIL",
  "purpose": "PROMOTIONAL",
  "listName": "MasterList",
  "htmlMessagePath": "/messagelibrary/email/1040/Message.htm",
  "textMessagePath": "/messagelibrary/email/1040/Message.itl.txt",
  "enableLinkTracking": false,
  "attachmentPaths": [
    "/contentlibrary/example/docs/my.htm"
  ],
  "enableExternalTracking": false,
  "subject": "EMD Camp",
  "fromName": "api.user",
  "fromEmail": "api.user@responsys.com",
  "replyToEmail": "api.user@responsys.com",
  "useUTF8": false,
  "locale": "en",
  "trackHTMLOpens": true,
  "trackConversions": false,
  "sendTextIfHTMLUnknown": false,
  "unsubscribeOption": "OPTOUT_SINGLE_CLICK",
  "autoCloseOption": "AUTO_CLOSE_X_DAYS_AFTER_LAST_RESPONSE",
  "autoCloseValue": "180",
  "links": [
    {
      "rel": "self",
      "href": "/rest/api/v1.3/campaigns/EMD Camp1",
      "method": "GET"
    },
    {
      "rel": "update",
      "href": "/rest/api/v1.3/campaigns/EMD Camp1",
      "method": "PUT"
    },
    {
      "rel": "create",
      "href": "/rest/api/v1.3/campaigns",
      "method": "POST"
    }
  ]
}
```

Get all campaigns

Obtain the campaign properties for all EMD Email, Push, Message Center, SMS, or MMS campaigns.

Important: To use the Get All Campaigns API to get all EMD email campaigns, your account must be enabled for Email Message Designer (EMD). Otherwise using the call will result in an error with HTTPS status code of 401 Unauthorized and API_DISABLED_FOR_USER in the error message payload.

Service URL:

`/rest/api/v1.3/campaigns`

Optional Path Parameters:

Tip: Leave the following parameters set to their default values and use the “prev” and “next” links returned in the response to get additional campaigns as needed.

- `offset`: starts at 0 and indicates the record number for the response result set (defaults to 0).
- `limit`: number of campaigns to return in the response (defaults to 200 and cannot exceed 200).
- `type`: For EMD email campaigns, this can be omitted or use `type=email`. For clearer client application code, we recommend including `type=email`, but we allow it to be omitted for backward compatibility.

Request Method:

```
GET
```

Request Header:

```
Authorization=<AUTH_TOKEN>

Content-Type=application/json
```

Request Body - Required Properties:

```
Not applicable
```

RESPONSE NOTES:

- The response contains an array of up to 200 EMD Email campaign objects per request. This API does not support fetching Classic Email campaigns, so the response will not contain Classic Email campaign objects.

- Each item in the array contains a campaign object, which includes the campaign ID, the campaign's other properties, and a links array containing the campaign object's related API operations (specific to the campaign name where applicable).
- The campaign properties returned in the response may vary, depending on what was configured for the campaign when it was created.
- Campaign operations links may or may not be supported by your level of access.
- The response also contains a links array for the "Get all Email campaigns" interface. This array contains the endpoints for obtaining the previous and next batch of campaign objects.

Sample Request Body

Not applicable

Sample Response Body

The endpoint `/rest/api/v1.3/campaigns?type=email&offset=0&limit=5` returned the following response (for brevity, the sample shows only the first and last campaign of the five returned.):

```
{
  "campaigns": [
    {
      "id": 1000,
      "name": "testcampaign-b11",
      "folderName": "testfolder",
      "type": "EMAIL",
      "description": "description",
      "purpose": "PROMOTIONAL",
      "marketingStrategy": "strategy",
      "marketingProgram": "program",
      "listName": "listname",
      "filterPaths": [
        "foldername/objectName1",
        "foldername/objectName2"
      ],
      "refiningDataSourcePath": "foldername/objectName1",
      "proofListPath": "foldername/objectName1",
      "seedListPath": "foldername/objectName1",
      "segmentPaths": [
        "foldername/objectName1",
        "foldername/objectName2"
      ],
      "supplementaryCampaignDataSourcePaths": [
        "foldername/objectName1",
        "foldername/objectName2"
      ],
      "supplementaryLookupDataSourcePaths"[
        "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/v1.3/campaigns/testcampaign-b11",
          "method": "GET"
        },
        {
          "rel": "create",
          "href": "/rest/api/v1.3/campaigns",
          "method": "POST"
        },
        {
          "rel": "updateCampaign",
          "href": "rest/api/v1.3/campaigns/testcampaign-b11",
          "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"
      ],
      "supplementaryLookupDataSourcePaths"[
        "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/v1.3/campaigns/testcampaign-b12",
          "method": "GET"
        },
        {
          "rel": "create",
          "href": "/rest/api/v1.3/campaigns",
          "method": "POST"
        },
        {
          "rel": "updateCampaign",
          "href": "rest/api/v1.3/campaigns/testcampaign-b12",
          "method": "PUT"
        }
      ]
    }
  ],
  "links": [
    {
      "rel": "self",
      "href": "/rest/api/v1.3/campaigns?type=email",
      "method": "GET"
    },
    {
      "rel": "prev",
      "href":
"/rest/api/v1.3/campaigns?limit=200&offset=0&type=email",
      "method": "GET"
    },
    {
      "rel": "next",
      "href":
"rest/api/v1.3/campaigns?limit=200&offset=200&type=email",
      "method": "GET"
    }
  ]
}
```

Sample Request URL for Push campaigns

```
/rest/api/v1.3/campaigns?type=push
```

Tip: Leave the following parameters set to their default values and use the "prev" and "next" links returned in the response to get additional Push campaigns as needed.

Sample Response

```
{
  "campaigns": [
    {
      "id": 1000,
      "name": "testcampaign-b11",
      "folderName": "testfolder",
      "type": "EMAIL",
      "description": "description",
      "purpose": "PROMOTIONAL",
      "marketingStrategy": "strategy",
      "marketingProgram": "program",
      "listName": "listname",
      "filterPaths": [
        "foldername/objectName1",
        "foldername/objectName2"
      ],
      "refiningDataSourcePath": "foldername/objectName1",
      "proofListPath": "foldername/objectName1",
      "seedListPath": "foldername/objectName1",
      "segmentPaths": [
        "foldername/objectName1",
        "foldername/objectName2"
      ],
      "supplementaryCampaignDataSourcePaths": [
        "foldername/objectName1",
        "foldername/objectName2"
      ],
      "supplementaryLookupDataSourcePaths"[
        "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/v1.3/campaigns/testcampaign-b11",
          "method": "GET"
        },
        {
          "rel": "create",
          "href": "/rest/api/v1.3/campaigns",
          "method": "POST"
        },
        {
          "rel": "updateCampaign",
          "href": "rest/api/v1.3/campaigns/testcampaign-b11",
          "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"
      ],
      "supplementaryLookupDataSourcePaths"[
        "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/v1.3/campaigns/testcampaign-b12",
          "method": "GET"
        },
        {
          "rel": "create",
          "href": "/rest/api/v1.3/campaigns",
          "method": "POST"
        },
        {
          "rel": "updateCampaign",
          "href": "rest/api/v1.3/campaigns/testcampaign-b12",
          "method": "PUT"
        }
      ]
    }
  ],
  "links": [
    {
      "rel": "self",
      "href": "/rest/api/v1.3/campaigns?type=email",
      "method": "GET"
    },
    {
      "rel": "prev",
      "href":
"/rest/api/v1.3/campaigns?limit=200&offset=0&type=email",
      "method": "GET"
    },
    {
      "rel": "next",
      "href":
"rest/api/v1.3/campaigns?limit=200&offset=200&type=email",
      "method": "GET"
    }
  ]
}
```

Sample Request URL for SMS campaigns

```
/rest/api/v1.3/campaigns?type=sms
```

Tip: Leave the following parameters set to their default values and use the "prev" and "next" links returned in the response to get additional SMS campaigns as needed.

Sample Response

The endpoint `/rest/api/v1.3/campaigns?type=sms` returned the following response (in this account, the limit set was for 5 campaigns, but there were only two SMS campaigns for the account):

```
{
  "campaigns": [
    {
      "id": 1234,
      "name": "SMS_CAMP1",
      "folderName": "Folder1",
      "type": "SMS",
      "purpose": "PROMOTIONAL",
      "listName": "R_SMS_LIST1",
      "textMessagePath": "/messagelibrary/sms/13187/Message.txt",
      "enableLinkTracking": false,
      "enableExternalTracking": false,
      "useUTF8": false,
      "trackHTMLOpens": false,
      "trackConversions": false,
      "sendTextIfHTMLUnknown": false,
      "unsubscribeOption": "OPTOUT_SINGLE_CLICK",
      "autoCloseOption": "AUTO_CLOSE_X_DAYS_AFTER_LAST_RESPONSE",
      "autoCloseValue": "90"
    },
    {
      "id": 12345,
      "name": "SMS_CAMP2",
      "folderName": "Folder2",
      "type": "SMS",
      "purpose": "PROMOTIONAL",
      "listName": "R_SMS_LIST2",
      "textMessagePath": "/messagelibrary/sms/13687/Message.txt",
      "enableLinkTracking": false,
      "enableExternalTracking": false,
      "campaignVariables": {
        "SMS_CARRIER": null,
        "SMS_USER_INPUT1": null,
        "SMS_USER_INPUT2": null,
        "SMS_USER_INPUT3": null,
        "SMS_USER_INPUT4": null,
        "SMS_USER_INPUT5": null,
        "SMS_CODE": null,
        "SMS_KEYWORD": " Default"
      },
      "useUTF8": false,
      "trackHTMLOpens": false,
      "trackConversions": false,
      "sendTextIfHTMLUnknown": false,
      "unsubscribeOption": "OPTOUT_SINGLE_CLICK",
      "autoCloseOption": "AUTO_CLOSE_X_DAYS_AFTER_LAST_RESPONSE",
      "autoCloseValue": "90"
    }
  ],
  "links": [
    {
      "rel": "self",
      "href": "/rest/api/v1.3/campaigns?type=sms",
      "method": "GET"
    },
    {
      "rel": "prev",
      "href": "/rest/api/v1.3/campaigns?limit=5&offset=0&type=sms",
      "method": "GET"
    },
    {
      "rel": "next",
      "href":
"rest/api/v1.3/campaigns?limit=5&offset=200&type=sms",
      "method": "GET"
    }
  ]
}
```

Sample Request URL for MMS campaigns

```
/rest/api/v1.3/campaigns?type=mms
```

Tip: Leave the following parameters set to their default values and use the "prev" and "next" links returned in the response to get additional MMS campaigns as needed.

Sample Response

```
{
    "campaigns": [
        {
            "id": 10016762,
            "name": "test-mms",
            "folderName": "examples",
            "type": "MmsCampaign",
            "purpose": "PROMOTIONAL",
            "textMessagePath":
"/messagelibrary/mms/10016762/Message.txt",
            "enableLinkTracking": false,
            "attachmentPaths": [
                null
            ],
            "enableExternalTracking": false,
            "useUTF8": false,
            "trackHTMLOpens": false,
            "trackConversions": false,
            "sendTextIfHTMLUnknown": false,
            "autoCloseOption": "None"
        },
                .
                .
                .
        {
            "id": 11081025,
            "name": "AG_AudienceTest_MMS",
            "folderName": "AG-Test",
            "type": "MmsCampaign",
            "purpose": "PROMOTIONAL",
            "listName": "2018 list",
            "textMessagePath":
"/messagelibrary/mms/11081025/Message.txt",
            "enableLinkTracking": false,
            "attachmentPaths": [
                "/contentlibrary/2018_content^sale70.jpg"
            ],
            "enableExternalTracking": false,
            "useUTF8": false,
            "trackHTMLOpens": false,
            "trackConversions": false,
            "sendTextIfHTMLUnknown": false,
            "autoCloseOption": "None"
        }
    ],
    "links": [
        {
            "rel": "self",
            "href": "/rest/api/v1.3/campaigns?type=mms",
            "method": "GET"
        }
    ]
}
```

Sample Request URL for Message Center:

```
/rest/api/v1.3/campaigns?type=pushiocampaign
```

Tip: Leave the following parameters set to their default values and use the "prev" and "next" links returned in the response to get additional Message Center campaigns as needed.

Sample Response

The response data for Message Center campaigns returns two attributes that are specific to this channel: `appName` contains the name of the mobile app, and `destinationMessageCenter` contains the name of the message center to which the marketer wanted the campaign message to be sent.

```
{
  "campaigns": [
    {
      "id": 11896453,
      "name": "jmp-test-mc-021419",
      "folderName": "jmp-test",
      "type": "PushIOCampaign",
      "purpose": "PROMOTIONAL",
      "marketingStrategy": "Other",
      "marketingProgram": "Other",
      "listName": "mmlist",
      "channelList": "mm_APP",
      "appName": "JMP Training App",
      "destinationMessageCenter": "Primary",
      "textMessagePath":
"/messagelibrary/push/11896453/Message.txt",
      "enableLinkTracking": false,
      "enableExternalTracking": false,
      "useUTF8": false,
      "trackHTMLOpens": false,
      "trackConversions": false,
      "sendTextIfHTMLUnknown": false
    }
  ],
  "links": [
    {
      "rel": "self",
      "href": "/rest/api/v1.3/campaigns?type=pushiocampaign",
      "method": "GET"
    }
  ]
}
```

Get all Push campaigns

Use this interface to get Push campaigns and their properties. For this API, “Push” includes all variations of Mobile App campaigns: Push, Push campaigns that also sends to Message Center, Rich Push, In-app Message, and Message Center direct campaigns.

Important: To use the Get all Campaigns API to obtain Push campaigns, your account must be enabled for Push. Otherwise using the call will result in an error (HTTPS status code 401 Unauthorized and error code API_DISABLED_FOR_USER).

Service URL:

`/rest/api/v1.3/campaigns?type=push`

Required Path Parameters:

`type`: For Push campaigns, you must include `type=push`.

Optional Path Parameters:

Tip: Leave the following parameters set to their default values and use the “prev” and “next” links returned in the response to get additional campaigns as needed.

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

Request Method:

`GET`

Request Header:

```
Authorization=<AUTH_TOKEN>

Content-Type=application/json
```

Request Body - Required Properties:

`Not applicable`

Sample Request Body

Not applicable

Response Body Notes:

- The response contains an array of up to 200 campaign objects per request.
- Each item in the array contains a campaign object, which includes the campaign ID and the campaign's other properties.
- The campaign properties returned in the response may vary, depending on what was configured for the campaign when it was created.
- The response also contains a links array for the "Get all Push campaigns" interface. This array contains the endpoints for obtaining the previous and next batch of campaign objects.
- The response body for Push includes some EMD campaign attributes, even though they are not available in the Push campaign workbook. These values are set either to "`false`" or to the account-level settings.

  Examples:

  ```
      "trackHTMLOpens": false,
      "trackConversions": false,
      "sendTextIfHTMLUnknown": false,
      "unsubscribeOption": "OPTOUT_SINGLE_CLICK",
      "autoCloseOption": "AUTO_CLOSE_X_DAYS_AFTER_LAUNCH",
      "autoCloseValue": "30"
  ```

- By default, Campaign Purpose is `Promotional` for Push.

- The response body for Push includes a property specific to all types of Mobile App campaigns called `channelList`. The value is the App Channel List name used by the campaign, and this property is only returned when `type=push`.

Sample Response Body

The endpoint `/rest/api/v1.3/campaigns?type=push&offset=0&limit=5` returned the following response (for brevity, the sample shows only the first and last campaign of the five returned.):

```
{
    "campaigns": [
        {
            "id": 13587,
            "name": "MyPushCampaign1",
            "folderName": "PushCampaigns",
            "type": "PUSH",
            "purpose": "PROMOTIONAL",
            "listName": "R_Profile_List1",
            "channelList": "R_Profile_List1_APP",
            "textMessagePath":
"/messagelibrary/push/13587/Message.txt",
            "enableLinkTracking": false,
            "enableExternalTracking": false,
            "useUTF8": false,
            "trackHTMLOpens": false,
            "trackConversions": false,
            "sendTextIfHTMLUnknown": false,
            "unsubscribeOption": "OPTOUT_SINGLE_CLICK",
            "autoCloseOption": "AUTO_CLOSE_X_DAYS_AFTER_LAST_
RESPONSE",
            "autoCloseValue": "90"
        },
        .
        .
        .
        {
            "id": 15647,
            "name": "PushCampaign5",
            "folderName": "MorePushCampaigns",
            "type": "PUSH",
            "purpose": "PROMOTIONAL",
            "listName": "R_Profile_List1",
            "channelList": "R_Profile_List1_APP",
            "textMessagePath":
"/messagelibrary/push/15647/Message.txt",
            "enableLinkTracking": false,
            "enableExternalTracking": false,
            "useUTF8": false,
            "trackHTMLOpens": false,
            "trackConversions": false,
            "sendTextIfHTMLUnknown": false,
            "unsubscribeOption": "OPTOUT_SINGLE_CLICK",
            "autoCloseOption": "AUTO_CLOSE_X_DAYS_AFTER_LAST_
RESPONSE",
            "autoCloseValue": "90"
        }
    ],
    "links": [
        {
            "rel": "self",
            "href":
"/rest/api/v1.3/campaigns?type=push&offset=0&limit=5",
            "method": "GET"
        },
        {
            "rel": "next",
            "href":
"/rest/api/v1.3/campaigns?limit=5&offset=5&type=push",
            "method": "GET"
        }
    ]
}
```

Get all Message Center campaigns

Use this interface to get Message Center campaigns and their properties.

Important: To use the Get all Campaigns API to obtain Message Center campaigns, your account must be enabled for Push and for Message Center. Otherwise using the call will result in an error (HTTPS status code 401 Unauthorized and error code API_DISABLED_FOR_USER).

Service URL:

`/rest/api/v1.3/campaigns?type=pushiocampaign`

Required Path Parameters:

`type`: For Message Center campaigns, you must include `type=pushiocampaign`.

Optional Path Parameters:

Tip: Leave the following parameters set to their default values and use the “prev” and “next” links returned in the response to get additional campaigns as needed.

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

Request Method:

`GET`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Request Body - Required Properties:

`Not applicable`

Sample Request Body

Not applicable

Response Body Notes:

- The response contains an array of up to 200 campaign objects per request.
- Each item in the array contains a campaign object, which includes the campaign ID and the campaign's other properties.
- The campaign properties returned in the response may vary, depending on what was configured for the campaign when it was created.
- The response also contains a links array for the "Get all Push campaigns" interface. This array contains the endpoints for obtaining the previous and next batch of campaign objects.
- The response body for Push includes some EMD campaign attributes, even though they are not available in the Message Center campaign workbook. These values are set either to "`false`" or to the account-level settings.

  Examples:

  ```
  "trackHTMLOpens": false,
  "trackConversions": false,
  "sendTextIfHTMLUnknown": false
  ```

- By default, Campaign Purpose is `Promotional` for Message Center.
- The response bodies for Push and for Message Center includes a property called `channelList`. The value is the App Channel List name used by the campaign, and this property is only returned when `type=push` or `type=pushiocampaign`.
- When you query for Message Center campaigns using the `type=pushiocampaign`, the response body returns two attributes that are specific to this channel: `appName` contains the name of the mobile app, and `destinationMessageCenter` contains the name of the message center to which the marketer wanted the campaign message to be sent.

Sample Response Body

For an account with one Message Center campaign, the endpoint `/rest/api/v1.3/campaigns?type=pushiocampaign` returned the following response:

```
{
  "campaigns": [
    {
      "id": 11896453,
      "name": "jmp-test-mc-021419",
      "folderName": "jmp-test",
      "type": "PushIOCampaign",
      "purpose": "PROMOTIONAL",
      "marketingStrategy": "Other",
      "marketingProgram": "Other",
      "listName": "mmlist",
      "channelList": "mm_APP",
      "appName": "JMP Training App",
      "destinationMessageCenter": "Primary",
      "textMessagePath":
"/messagelibrary/push/11896453/Message.txt",
      "enableLinkTracking": false,
      "enableExternalTracking": false,
      "useUTF8": false,
      "trackHTMLOpens": false,
      "trackConversions": false,
      "sendTextIfHTMLUnknown": false
    }
  ],
  "links": [
    {
      "rel": "self",
      "href": "/rest/api/v1.3/campaigns?type=pushiocampaign",
      "method": "GET"
    }
  ]
}
```

Get all SMS campaigns

Use this interface to get SMS all campaigns and their properties.

Important: To use the Get All Campaigns API to obtain SMS campaigns, your account must be enabled for SMS. Otherwise using the call will result in an error (HTTPS status code 401 Unauthorized and error code API_DISABLED_FOR_USER).

Service URL:

`/rest/api/v1.3/campaigns?type=sms`

Required Path Parameters:

`type`: For SMS campaigns, you must include `type=sms`.

Optional Path Parameters:

Tip: Leave the following parameters set to their default values and use the “prev” and “next” links returned in the response to get additional campaigns as needed.

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

Request Method:

`GET`

Request Header:

```
Authorization=<AUTH_TOKEN>

Content-Type=application/json
```

Request Body - Required Properties:

`Not applicable`

Sample Request Body

Not applicable

Response Body Notes:

- The response contains an array of up to 200 campaign objects per request.
- Each item in the array contains a campaign object, which includes the campaign ID and the campaign's other properties.
- The campaign properties returned in the response may vary, depending on what was configured for the campaign when it was created.
- The response also contains a links array for the "Get all campaigns" interface for SMS campaign objects. This array contains the endpoints for obtaining the previous and next batch of campaign objects.
- The response body for SMS includes some EMD campaign attributes, even though they are not available in the SMS campaign workbook. These values are set either to "false" or to the account-level settings.

  Examples:

  ```
  "trackHTMLOpens": false,
  "trackConversions": false,
  "sendTextIfHTMLUnknown": false,
  "unsubscribeOption": "OPTOUT_SINGLE_CLICK",
  "autoCloseOption": "AUTO_CLOSE_X_DAYS_AFTER_LAUNCH",
  "autoCloseValue": "30"
  ```

Sample Response Body

The endpoint `/rest/api/v1.3/campaigns?type=sms&offset=0&limit=5` returned the following response (for brevity, the sample shows only the first and last campaign of the five returned.):

```
{
    "campaigns": [
        {
            "id": 1234,
            "name": "SMS_CAMP1",
            "folderName": "Folder1",
            "type": "SMS",
            "purpose": "PROMOTIONAL",
            "listName": "R_SMS_LIST1",
            "textMessagePath":
"/messagelibrary/sms/13187/Message.txt",
            "enableLinkTracking": false,
            "enableExternalTracking": false,
            "useUTF8": false,
            "trackHTMLOpens": false,
            "trackConversions": false,
            "sendTextIfHTMLUnknown": false,
            "unsubscribeOption": "OPTOUT_SINGLE_CLICK",
            "autoCloseOption": "AUTO_CLOSE_X_DAYS_AFTER_LAST_
RESPONSE",
            "autoCloseValue": "90"
        },
        {
            "id": 12345,
            "name": "SMS_CAMP2",
            "folderName": "Folder2",
            "type": "SMS",
            "purpose": "PROMOTIONAL",
            "listName": "R_SMS_LIST2",
            "textMessagePath":
"/messagelibrary/sms/13687/Message.txt",
            "enableLinkTracking": false,
            "enableExternalTracking": false,
            "campaignVariables": {
                "SMS_CARRIER": null,
                "SMS_USER_INPUT1": null,
                "SMS_USER_INPUT2": null,
                "SMS_USER_INPUT3": null,
                "SMS_USER_INPUT4": null,
                "SMS_USER_INPUT5": null,
                "SMS_CODE": null,
                "SMS_KEYWORD": " Default"
            },
            "useUTF8": false,
            "trackHTMLOpens": false,
            "trackConversions": false,
            "sendTextIfHTMLUnknown": false,
            "unsubscribeOption": "OPTOUT_SINGLE_CLICK",
            "autoCloseOption": "AUTO_CLOSE_X_DAYS_AFTER_LAST_
RESPONSE",
            "autoCloseValue": "90"
        }
    ]
    "links": [
            {
                "rel": "self",
                "href":
"/rest/api/v1.3/campaigns?type=sms&offset=0&limit=5",
                "method": "GET"
            },
            {
                "rel": "next",
                "href":
"/rest/api/v1.3/campaigns?limit=5&offset=5&type=sms",
                "method": "GET"
            }
        ]

}
```

Get all MMS campaigns

Use this interface to get MMS all campaigns and their properties.

Important: To use the Get All Campaigns API to obtain MMS campaigns, your account must be enabled for MMS. Otherwise using the call will result in an error

(HTTPS status code 401 Unauthorized and error code API_DISABLED_FOR_USER).

Service URL:

`/rest/api/v1.3/campaigns?type=mms`

Required Path Parameters:

`type`: For MMS campaigns, you must include `type=mms`.

Optional Path Parameters:

Tip: Leave the following parameters set to their default values and use the “prev” and “next” links returned in the response to get additional campaigns as needed.

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

Request Method:

`GET`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Request Body - Required Properties:

`Not applicable`

Sample Request Body

Not applicable

Response Body Notes:

- The response contains an array of up to 200 campaign objects per request.
- Each item in the array contains a campaign object, which includes the campaign ID and the campaign's other properties.
- The campaign properties returned in the response may vary, depending on what was configured for the campaign when it was created.
- The response also contains a links array for the "Get all campaigns" interface for MMS campaign objects. This array contains the endpoints for obtaining the previous and next batch of campaign objects.
- The response body for MMS includes some EMD campaign attributes, even though they are not available in the MMS campaign workbook. These values are set either to "false" or to the account-level settings.

  Examples:

  ```
  "trackHTMLOpens": false,
  "trackConversions": false,
  "sendTextIfHTMLUnknown": false
  ```

Sample Response Body

The endpoint `/rest/api/v1.3/campaigns?type=mms` returned the following response (for brevity, the sample shows only the first and last campaigns of the full list of campaigns returned.):

```
{
  "campaigns": [
    {
```

```
      "id": 10016762,
      "name": "test-mms",
      "folderName": "examples",
      "type": "MmsCampaign",
      "purpose": "PROMOTIONAL",
      "textMessagePath":
"/messagelibrary/mms/10016762/Message.txt",
      "enableLinkTracking": false,
      "attachmentPaths": [
        null
      ],
      "enableExternalTracking": false,
      "useUTF8": false,
      "trackHTMLOpens": false,
      "trackConversions": false,
      "sendTextIfHTMLUnknown": false,
      "autoCloseOption": "None"
    },
    .
    .
    .
    {
      "id": 11081025,
      "name": "AG_AudienceTest_MMS",
      "folderName": "AG-Test",
      "type": "MmsCampaign",
      "purpose": "PROMOTIONAL",
      "listName": "2018 list",
      "textMessagePath":
"/messagelibrary/mms/11081025/Message.txt",
      "enableLinkTracking": false,
      "attachmentPaths": [
        "/contentlibrary/2018_content^sale70.jpg"
      ],
      "enableExternalTracking": false,
      "useUTF8": false,
      "trackHTMLOpens": false,
      "trackConversions": false,
      "sendTextIfHTMLUnknown": false,
      "autoCloseOption": "None"
    }
  ],
  "links": [
    {
      "rel": "self",
      "href": "/rest/api/v1.3/campaigns?type=mms",
      "method": "GET"
      }
    ]
}
```

Programs

Get all programs

Publish or Unpublish a Program

Get all programs

Use this interface to get a list of Responsys program orchestrations for an account and the associated metadata for each program. The response includes draft and published programs, and it includes program status information.

Service URL:

`/rest/api/v1.3/programs`

Required Path Parameters:

None

Optional Path Parameters:

Tip: Leave the following parameters set to their default values and use the “prev” and “next” links returned in the response to get additional programs as needed.

- `offset`: starts at 0 and indicates the record number for the response result set (defaults to 0).
- `limit`: number of programs to return in the response (defaults to 200 and cannot exceed 200).
- `status`: comma separated list of status of programs to be fetched. Allowed values are `NOT_RUNNING`, `RUNNING`, `RUNNING_WITH_ERRORS`.

Request Method:

`GET`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Request Body - Required Properties:

`Not applicable`

RESPONSE NOTES:

- The response contains an array of up to 200 program objects per request.
- The response also contains a links array for the “Get all Programs” interface. This array contains the endpoints for obtaining the previous and next batch of program objects.

Sample Request Body

Not applicable

Sample Response Body

The endpoint `/rest/api/v1.3/programs` returned the following response (for brevity, the sample shows only a sample of the 200 returned.):

```
{
  "programs": [
    {
      "id": 10001362,
      "name": "PD_100_Reactivation",
      "folderName": "PD_100",
      "listName": "PD_100_LIST",
      "channels": [
        "EMAIL"
      ],
      "createdOn": "07/10/2018",
      "createdBy": "N/A",
      "modifiedOn": "07/10/2018",
      "modifiedBy": "N/A",
      "status": "NOT_RUNNING"
    },
    {
      "id": 10004423,
      "name": "Simple Program",
      "folderName": "Examples",
      "listName": "Example contact list",
      "channels": [],
      "createdOn": "29/02/2018",
      "createdBy": "N/A",
      "modifiedOn": "29/02/2018",
      "modifiedBy": "N/A",
      "publishDate": "29/02/2018",
      "unpublishDate": "30/05/2018",
      "status": "NOT_RUNNING"
    },
    {
      "id": 10021809,
      "name": "OOWProgram2",
      "folderName": "Examples",
      "listName": "TC_Food_List",
      "channels": [
        "EMAIL",
        "MOBILE",
        "MOBILE_APP"
      ],
      "createdOn": "08/09/2018",
      "createdBy": "Admin",
      "modifiedOn": "21/09/2018",
      "modifiedBy": "Admin",
      "publishDate": "21/09/2018",
      "unpublishDate": "21/12/2018",
      "status": "NOT_RUNNING"
    },
    .
```

```
     .
     .
     {
       "id": 10025948,
       "name": "Summer Deals - In App",
       "folderName": "Examples",
       "listName": "2018 list",
       "channels": [],
       "createdOn": "19/10/2018",
       "createdBy": "Admin",
       "modifiedOn": "19/10/2018",
       "modifiedBy": "Admin",
       "publishDate": "19/10/2018",
       "status": "RUNNING"
     }
   ],
   "links": [
     {
       "rel": "self",
       "href": "/rest/api/v1.3/programs",
       "method": "GET"
     },
     {
       "rel": "next",
       "href": "/rest/api/v1.3/programs?limit=200&offset=200",
       "method": "GET"
     }
   ]
 }
```

Publish or Unpublish a Program

Use this interface to publish a valid program or unpublish a program.

Service URL:

`/rest/api/v1.4/{programName}`

Required Path Parameters:

`programName` - Name of the Program to be published or unpublished.

Request Method:

`PATCH`

Request Header:

```
Authorization=<AUTH_TOKEN>
```

```
Content-Type=application/json
```

Request Body Properties:

`action` - Action to be performed on a given program. Allowed values include `publish` and `unpublish`.

`saveDraft` - Indicates whether to save the draft version or published version. Allowed values include `Y` and `N`.

Sample Request Body

Publish a program.

```
{
  "action": "publish",
  "saveDraft": "Y"
}
```

Sample Response Body - Success

```
{
  "status": "SUCCESS",
  "errorMsg": []
}
```

Sample Response Body - Failure

```
{
  "type": "",
  "title": "Invalid request parameters",
  "errorCode": "INVALID_PARAMETER",
  "detail": "Request parameter 'action' should be set as
'unpublish' or 'publish'",
  "errorDetails": []
}
```

Triggering Email Messages

Responsys can send personalized email messages to email addresses.

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

Use the following interface to merge members into a profile list and subsequently send them an email message.

REQUEST NOTES:

- You can send Responsys Email campaigns that already exist to up to 200 members of a profile list.
- For the list associated with the specified Email campaign, you must specify an array of record data that contain field names and their corresponding field values.
- Do not use system column names for the optional data, such as `EMAIL_ADDRESS_`. This may cause incorrect data to be processed, possibly resulting in skipped records and incorrect data being recorded in the Event table.
- To pass extended/accented characters in `optionalData` payload, they must be escaped as Unicode characters. For example, the euro symbol € is escaped as `\u20AC`, the yen symbol ¥ is escaped as `\u00A5`, an ü is escaped as `\u00FC`, an é is escaped as `\u00E9`, and the like. Otherwise, you may receive an `INVALID_REQUEST_CONTENT` error.

Please see the definition of merge rule parameters provided in Definitions of Rule Parameters for Merging Members into a Profile List.

Service URL:

```
/rest/api/v1.3/campaigns/{campaignName}/email
```

Required Path Parameter:

```
campaignName
```

Request Method:

```
POST
```

Request Header:

```
Authorization=<AUTH_TOKEN>

Content-Type=application/json
```

Sample Request Body:

```
{
   "mergeTriggerRecordData":{
      "mergeTriggerRecords":[
         {
            "fieldValues":[
               "mdi1234@foobar.com",
               "martiness"
            ],
            "optionalData":[
               { "name":"FIRST_NAME", "value":"jim_1" },
               { "name":"LAST_NAME", "value":"smith_1" }
            ]
         },
         {
            "fieldValues":[
               "mdi.1234@foobarcorp.com",
               "concord"
            ],
            "optionalData":[
               { "name":"FIRST_NAME", "value":"jim_2" },
               { "name":"LAST_NAME", "value":"smith_2" }
            ]
         }
      ],
      "fieldNames":[
         "EMAIL_ADDRESS_",
         "CITY_"
      ]
   },
   "mergeRule": {
      "htmlValue":"H",
      "matchColumnName1":"EMAIL_ADDRESS_",
      "matchColumnName2":null,
      "optoutValue":"O",
      "insertOnNoMatch":true,
      "defaultPermissionStatus":"OPTIN",
      "rejectRecordIfChannelEmpty":"E",
      "optinValue":"I",
      "updateOnMatch":"REPLACE_ALL",
      "textValue":"T",
      "matchOperator":"NONE"
   }
}
```

Sample Response:

The response returns a status for each record sent in the request. In this example, the first record was successfully merged and the email was sent to the recipient whose RIID is 72067. The second record was successfully merged, but the email was undeliverable.

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

Merge members into a profile list and trigger email messages with attachments

Use the following interface to merge members into a profile list and subsequently send them an email message with one or more attached files.

Important: This is a controlled feature. To have this API enabled for your account, contact your Customer Success Manager (CSM).

REQUEST NOTES:

- The API payload can trigger each batched recipient with their own attachment(s), or with no attachments.

  To send an email campaign to a recipient that does not include attachments, omit the `attachmentData` array from the recipient object. (If you send null values in the `attachmentData` array, the email for the recipient fails with the invalid attachment type error message.)

- Each attachment must be Base-64 encoded in the API payload. Any personalization of *attachments* must happen before your client application includes the attachment in the API payload.

- Default allowed file types are JPEG and JPG, PDF, and ICAL. Ensure that the extension is the correct type for the decoded file, otherwise the recipient will receive the attachment but not be able to open it.

- If attachments are already configured in the campaign that the API request will trigger, the email sent will include those attachments plus the attachments sent in the API request.
  - Each email can contain a maximum of 10 attachments total (campaign + API payload).
  - Total size of all attachments for a single email is limited to 500 KB.
  - Total number of emails sent in one hour is limited to 5000.

- Data attached by the API are scanned before the system creates the files on the server.

- For the list associated with the specified Email campaign, you must specify an array of record data that contain field names and their corresponding field values.

- Do not use system column names for the optional data, such as `EMAIL_ADDRESS_`. This may cause incorrect data to be processed, possibly resulting in skipped records and incorrect data being recorded in the Event table.
- To pass extended/accented characters in `optionalData` payload, they must be escaped as Unicode characters. For example, the euro symbol € is escaped as `\u20AC`, the yen symbol ¥ is escaped as `\u00A5`, an ü is escaped as `\u00FC`, an é is escaped as `\u00E9`, and the like. Otherwise, you may receive an `INVALID_REQUEST_CONTENT` error.

As with Merge Trigger Email, the following also apply to Merge Trigger Email with Attachments:

- You can send Responsys Email campaigns that already exist to up to 200 members of a profile list.
- For the list associated with the specified Email campaign, you must specify an array of record data that contain field names and their corresponding field values.
- Do not use system column names for the optional data, such as `EMAIL_ADDRESS_`. This may cause incorrect data to be processed, possibly resulting in skipped records and incorrect data being recorded in the Event table.
- To pass extended/accented characters in `optionalData` payload, they must be escaped as Unicode characters. For example, the euro symbol € is escaped as `\u20AC`, the yen symbol ¥ is escaped as `\u00A5`, an ü is escaped as `\u00FC`, an é is escaped as `\u00E9`, and the like. Otherwise, you may receive an `INVALID_REQUEST_CONTENT` error.

Please see the definition of merge rule parameters provided in Definitions of Rule Parameters for Merging Members into a Profile List.

Service URL:

`/rest/api/v1.3/campaigns/{campaignName}/emailAttachments`

Required Path Parameter:

`campaignName`

Request Method:

```
POST
```

Request Header:

```
Authorization=<AUTH_TOKEN>

Content-Type=application/json
```

Sample Request Body:

```
{
    "mergeTriggerRecordDataWithAttachments": {
      "mergeTriggerRecordsWithAttachments": [
        {
          "fieldValues": [
            "1",
            "mdi1124@barcorp.com"
          ],
          "optionalData": [
            {
              "name": "FIRST_NAME",
              "value": "Jane"
            },
            {
              "name": "LAST_NAME",
              "value": "Jones"
            }
          ],
          "attachmentData": [
            {
              "name": "Hello World.pdf",
              "value":
"JVBERi0xLjQKJYCAgIAKMSAwIG9iago8PC9QYWdlcyAyIDAgUiAvVmlld2VyUHJlZm
VyZW5jZXMgOCAwIFIgL1R5cGUgL0NhdGFsb2cgPj4KZW5kb2JqCjIgMCBvYmoKPDwvQ
291bnQgMSAvTWVkaWFCb3ggWzAgMCA1OTYgODQyIF0gL1R5cGUgL1BhZ2VzIC9SZXNv
dXJjZXMgMyAwIFIgL0tpZHMgWzUgMCBSIF0gPj4KZW5kb2JqCjMgMCBvYmoKPDwvRm9
udCA3IDAgUiA+PgplbmRvYmoKNCAwIG9iago8PC9GaWx0ZXIgL0ZsYXRlRGVjb2RlIC
9MZW5ndGggNjAgPj4Kc3RyZWFtCnic0zdUMDZSCEnjcgrhMlQwAEKwgLm5sZ6FuZGBo
UJILpeGR2pOTr5CeH5RToqipkJIFpdrCBcAWMsNBgplbmRzdHJlYW0KZW5kb2JqCjUg
MCBvYmoKPDwvUGFyZW50IDIgMCBSIC9UeXBlIC9QYWdlIC9Db250ZW50cyA0IDAgUiA
+PgplbmRvYmoKNiAwIG9iago8PC9CYXNlRm9udCAvQ291cmllci1Cb2xkIC9TdWJ0eX
BlIC9UeXBlMSAvRW5jb2RpbmcgL1dpbkFuc2lFbmNvZGluZyAvVHlwZSAvRm9udCA+P
```

```
gplbmRvYmoKNyAwIG9iago8PC8xIDYgMCBSID4+CmVuZG9iago4IDAgb2JqCjw8L0Rp
c3BsYXlEb2NUaXRsZSB0cnVlID4+CmVuZG9iago5IDAgb2JqCjw8L1N1YmplY3QgKP7
/AFMAYQBtAHAAbABlACAAYQBiAG8AdQB0ACAAYQAgAHMAaQBtAHAAbABlACAAJwBoAG
UAbABsAG8AIAB3AG8AcgBsAGQAJwAgAHUAcwBpAG4AZwAgAFAARABGACAAQwBsAG8Ad
wBuKSAvQ3JlYXRvciAo/v8AbwByAGcALgBwAGQAZgBjAGwAbwB3AG4ALgBzAGEAbQBw
AGwAZQBzAC4AYwBsAGkALgBIAGUAbABsAG8AVwBvAHIAbABkAFMAYQBtAHAAbABlKSA
vQXV0aG9yICj+/wBTAHQAZQBmAGEAbgBvACAAQwBoAGkAegB6AG8AbABpAG4AaSkgL0
NyZWF0aW9uRGF0ZSAoRDoyMDExMDMwNTIwMDYxNSswMScwMCcpIC9Qcm9kdWNlciAo/
v8AUABEAEYAIABDAGwAbwB3AG4AIABmAG8AcgAgAEoAYQB2AGEAIAAwAC4AMQAuADAp
IC9UaXRsZSAo/v8AUABEAEYAIABDAGwAbwB3AG4AIAAtACAASABlAGwAbABvACAAdwB
vAHIAbABkACAAcwBhAG0AcABsAGUpID4+CmVuZG9iagp4cmVmCjAgMTAKMDAwMDAwMD
AwMCA2NTUzNSBmDQowMDAwMDAwMDE1IDAwMDAwIG4NCjAwMDAwMDAwODggMDAwMDAgb
g0KMDAwMDAwMDE4NyAwMDAwMCBuDQowMDAwMDAwMjE5IDAwMDAwIG4NCjAwMDAwMDAz
NDkgMDAwMDAgbg0KMDAwMDAwMDQxMSAwMDAwMCBuDQowMDAwMDAwNTEwIDAwMDAwIG4
NCjAwMDAwMDA1MzkgMDAwMDAgbg0KMDAwMDAwMDU4MSAwMDAwMCBuDQp0cmFpbGVyCj
w8L1Jvb3QgMSAwIFIgL0luZm8gOSAwIFIgL1NpemUgMTAgPj4Kc3RhcnR4cmVmCjEwM
zcKJSVFT0Y="
            },
            {
              "name": "weather.jpg",
              "value":
"/9j/4AAQSkZJRgABAQEAYABgAAD/4QA6RXhpZgAATU0AKgAAAAgAA1EQAAEAAAABAQ
AAAFERAAQAAAABAAAAAFESAAQAAAABAAAAAAAAAAD/2wBDAAIBAQIBAQICAgICAgICA
wUDAwMDAwYEBAMFBwYHBwcGBwcICQsJCAgKCAcHCg0KCgsMDAwMBwkODw0MDgsMDAz/
2wBDAQICAgMDAwYDAwYMCAcIDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAw
MDAwMDAwMDAwMDAwMDAwMDAz/wAARCACtANIDASIAAhEBAxEB/8QAHwAAAQUBAQEBAQ
EAAAAAAAAAAAECAwQFBgcICQoL/8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhM
UEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RF
RkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTlJWWl5iZmqKjpKW
mp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl5ufo6erx8vP09fb3+P
n6/8QAHwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoL/8QAtREAAgECBAQDBAcFB
AQAAQJ3AAECAxEEBSExBhJBUQdhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYkNOEl8RcY
GRomJygpKjU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6goOEhYa
HiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4u
Pk5ebn6Onq8vP09fb3+Pn6/9oADAMBAAIRAxEAPwDIkXA4I+lM34Wq6NIW+bP4U4wtI
fl4r6A8cknnh8n5l+Y1msMH057ir9vo8k0mSP1qxPpOF2sq/UdqCdWZCzYOM++M1Ygv
vJPtS3Nh5Jx69DVWaB4m60C1RpxavGV+b86e+pLnaprHJwfypaB8xrfb17sBUkN/Co+
8p9s1i0UBzG1dartT5JAPbNUj4gZThmH51Slfy0z1xVOO2kuD8q9TihRDmZ01pd/a8t
zRMSW5o0C1+w2P7w/Ofxq4bbzl3ClcopC0ZxkLSrbEnGK1bWEN8vyj1q7Fo6kE7S3qf
SlzAcnPPpuX3dPrVd4GMm3+H2rrpNFWQH7tU73RVj6df50cwwuUw7fSGmb2qx/YhTGPpV
2GzaKT5e3Wp/KkmbkcLRcLGWbeZ3ND9B7ngT0lINjE/s//AHaK1DYyZ+8aKq4GXnaKFuljbletPERePvUTW
RPILbvXFMC/aXYY8AVe8mO5Tp+IrGSBomrT0y6ZeGx+6ZeGx+VS0A86CrpkNu/mKzbnQmlkI6
j2FdVavG6LuGB6igsu8jr6UrsDjZfE1GXR3Q9DXcCISt91R68OtSt91R68VHNaRlfmVafL
lOFbT5B0FQyQPQMo+tdgkA0FQyQPQMo+tdgbANG7Kq8dsVVttDF0vAxVuTVtXjZYiNo
```

```
4rpL/wtGCu0fN71JZeGti58n73enzBymXbHe22ti2td8KkAe9OXRFDDcAh71rWGkxxK
o4LHj61LZRkrp7QjIx9KuWbyH725WzjPrW9a6aoI6MT6dK0otGhYfwt9AKjmK5Tk1Rn
fvnPNNnsfNAyjGuw/sOA4JU59sf4UjaFCw/H07Ucwcpw7acAcKv3u1T2thJG2Ntdh/Y
MQZj3PTjpUn9mqPxGOe3+eKOYOU5m30ppV9cc9etX7LQMj7u76dx9a2V01FLe/T0FTk
AKf4R37UcxRkw6ZGh6qGXPXk8e1S5WItj5ufvY3f57mn3BiCybW6nIA7/5/wA9axtWv
5YWO1h83I5pbgQ+JNfWzjbHJPPBzmuO1Lxc0x2suNvTb2rT1p2ul+asC60rzM/L+Nax
SM5NkR8TEHr/AOPUVEfDLOd3ltzz1oqtCdTXht/LY/yqUKD2/Srkdvubqo/pTJrFg9Z
3KIdilcj73enQW2H/ABpwUDnO1unFC3iRNyd1AFpI2RdvZulAR1k+9j8Kal2twy9h04
qYMsT7t27FAFiKNpOMbj3NTS2LSDkq39adYHcrEMCtWEKl9rNtoKSM90wSpqnLbyQS7
lbjvWtf239zB68gVlyxzRg8sB0xQSy1BeoUxI3HHBNWJNUjgh6HGPzrBnZk/l0pqzyE
9/pQO5pf2n9om/xrQiljtkDK2T+prnd5LZBx9ad9uc9TkelAjrrHUhMPmAUdj6VqWur
wxJtbI/H+nauBg1JoDwzVYTxLnqefpS5SuY7ptV3q21e3B3daqzak1sFbdIf1/wDrVy
S+ImRdoZtvUjNMl1+R4yob9KXKHMdhF4kVwP3kYyP4hio5vEuG+V48e3+fr+VcK93Jy
evfrTP7TZRt3H1p8ocx2d34iaePHmYXvWe+v+Wfvtj3Ncy+qsW+9UM+ojZ96nyhzM6Y
+I4y3zSD06niql1rqN91g34VzL6nj3qFbmS4bjv6Cq5SeY2bzWVl68/hVeLW9r/LHu+
opun6S1y3OcfyroNK8LQy/wAYz70aAY/9sSH/AJZ0V2qeAI3RWDL8wz0oqeZFcrMFrL
byg+tRPas69DWkJoxGDyx9P/r1XNwkbfMehoJMue1kB+63NUpYih/xFdPBPBE3O3t9f5
1Jc6PBfR+ZGyKepU0XsOxysE7Lx8yirkV0ykcbtvar1x4ZBJZfToO9U301oPu02Imh1
J4WOON34VYk1Tz8H5QR3z0qjFEDuLjmoGfa+VHHWkBs2d9IT69xk9adcEz99v0rF+3M
I/lp1tqjSH5udvrQBfe1SY7d3O3ekbTwicfNTY7wJ3pBq6huduKAEa2wNrD36VDJZj
Hy1Yl1ONh71Wn1SJFJ3c0ANEIT71Jsj3Vn3ut4b5fWq66uyty351XKwNhpI0TpUb3gR
eNvtWXLrygDvzioLnWcp8vy+57UcormhPdST/1qBpCgwWX86x31Znk+U/8A16ltLWe7
P3QfTFVawuYtTXCA8uT7CoHuQ38J/OtS08E3F0itj8h0q5aeC1jf95J93jGKV0GpzbT
t/wDqqxaXTId1dNa+GLcllbmQdAB+tT2fguFX3MGIHqMfhS5g5WY9lrEh+7+lbO1X7M
y5jZsH0rSh0Kzgk3SKu0HgetbVjNaW6fu44dv1xmpkzTlII7+Qxr/o8h46+tFWj4gYH
7zf98iipKPPE1dj1P60Pq285/yaxt7RtQZ2Na8qMeY24NW8k7lY/wA6sR60R/F19a51
blhUiXXNLlHc6iPW933vmx056VH/AGhvb/e4zXPi5x/FQbtl5Bb8KOUZ0TSxiJvmG49
qoXEyqTVAXz4+9UckzSmjlAsG9YH5e9ENx5U25s7sVTO5R8tCxyEZz+FUA691lt/ysf
wqrJqjE7mbAqOeNhJ83H1qrdQEpkE8c4JoRF2WZ9fbop3VeirsIuPdCM/7PercC2nlnoFz7Voa
XAQ3rnoD3o5hooW2k7F3Z+pPPbT71LUhcikj0eS71Kn89qnPh8wScj7r/9xm7H4
jaKzCq2CV6jvWVJ4gYyZbjd3pxRo4toXt1zVC8tJJl+VevWpWpQH9Vzn9adN4gZN
wUlUb0rMj0i4aEERsfSopLG4+7s560yRg+hqOXXkkxTZHu57e9Z76Qzs
26ShLFbYcDc2OtPlQtS5/wkVyVg9ZhnnYNX3l6jH51Xug
XU/0oAz5HKHFNWY/X0p0sbBzxUYXH+NBmSeefpTln/Cocc0bS38qANCFxJViOLceKy7
eRo5OK0o5mU/WgtMtJppcDD0qSKxW3J3dueaba3zR/hRLe+c3v61Oyuh+ZeZB2FZL
wMzEbfwrYnk8xdvpRa2SkbjVCsYUGiPdT7V4Hf2q7B4R+f5mP41uWkUcEu7oRUk8oYH
+L8KXMxcpUj0BY48IFt235cY60B9vOanmZRVOirHjPNXrDSlXa23p
QsigZcj2ol1byeFb2HFGoG1B5OxlMSg4zuHpVS5thcHGMc4AzVO01rL5Y4PqO9Tvq0c
nVufXikMcnhvLfe5PpUp0BYju3bvr0qAasyNwabJrLSE/xfjQGhba9jsYdqj6c/8A1q
57WNWaST7v6VZklaSVmx+GantNKu3DKi9epoEtnNKu5tyA9jTpNO+XK6e+XO7PsK6j+wM
na7fNnGBWhp3hiWhCDS5SuJ5Ga8Sh+WdFsamVUP/AHqK9SGgLpe0Hynm8tmUPQMB
UEyxr1Vh7Zq4WUj5i2ap3gw36g1SZJTnVT/CfaqrKrHpVyDJb9TVEyREwTZ0596i
Zg33R79KkaPccEU0RbfXOc0hGmH3NVozqqDFVW3dqaXZh92gy8A/wCumNqCjp
VTymJpTDnuaB8zOlDLVKt8/aqRFW7Vbgb50lAK49L1bnNTRakSOtQlAA/7IoKL
i6mxjK7jtqGTUWJ71CoaVY80APN4z/xGo7x3rBsAcVOt8/x7UARx3zBsAcVOt8
```

```
4pgt1h4prpzQBa+2MelSRSSPVWI7StXIZgo/+tSAuWUbFwf0ratbl1QDCrj1NYkNyFq
cX7GoA2hcrjnj9ani1XYeu7noQc1hDUcDn5qeNTUfwgUFcx0a62+0bd23HHz0Vzv9qn
+83/fVFLlQcxhrIy0jkSLgj8qg8xgOtHmMRWpJIluPrTpbFW6cfWoxNQ03pQBG9jz2p
Hssr608ynHWk3n1oAg+xKDSi2X+7UuaCMigCFrZVPFOMeRjFSLHxTvIPtQBXWHHb9Kk
WLI+7Unlle2acNoPdaAIxDtGcU1k54p559acsJPXigCNV2inxpuPtTm2ofu5prS0ASC
NY+9BkFQ7qKAHMSxppTNG6jdQA7YoHX605X2Dv+NR7qC9AFhJcNxmpBP7VTWTJpwmIo
sBbE+TzT1fjg1TE2KBNU8oFzzv9r9aKqeePeijlAioooqgAHFGc0ZozQAoNJRmjNADh
SqVHUUwmjNAEnm4HAAppdj+NNooAdvOP/r0FvWm0UALnmkzRRQAUUUUAIRS4oooACM0
YoooAay5oCU6igAAoK8UZooAVV3Uvln/AD2puaM0AFFFFAEeaM0zcaXeaAHZopm40bj
QA+imbjS7zQA6jNN30m6gCTdSh6i3GjcaAJN9LvqPeaTcaAJd9G/iot5pd5oAk30b6i
3mjfQBLvo31FuNG6gCXfRvqLdRuNAEu+jfUW80bjQBLvo31FuNLvoAk30b6i3mjcaAJ
NxoqPcaKAIPNbFHmtTaKDMcJGAo8xvWm0Zo81sdabRQA7zWWoMiMe9NooAcZGY3rTaKA
Me9NooAcZGNHmNjrTaKAuOEjCjzGPem0UAO3mjea5etNooAd5jY60bzTaKAHeYfWjeab
jZptFADhIw70vmsKbRQAHeSw67daeg60/igLn/9k="
            }
          ]
        },
        {
          "fieldValues": [
            "2",
            "mdi.1234@foobarcorp.com"
          ],
          "optionalData": [
            {
              "name": "FIRST_NAME",
              "value": "Jim"
            },
            {
              "name": "LAST_NAME",
              "value": "Smith"
            }
          ],
          "attachmentData": [
            {
              "name": "Hello.pdf",
              "value":
"JVBERi0xLjQKJYCAgIAKMSAwIG9iago8PC9QYWdlcyAyIDAgUiAvVmlld2VyUHJlZm
VyZW5jZXMgOCAwIFIgL1R5cGUgL0NhdGFsb2cgPj4KZW5kb2JqCjIgMCBvYmoKPDwvQ
291bnQgMSAvTWVkaWFCb3ggWzAgMCA1OTYgODQyIF0gL1R5cGUgL1BhZ2VzIC9SZXNv
dXJjZXMgMyAwIFIgL0tpZHMgWzUgMCBSIF0gPj4KZW5kb2JqCjMgMCBvYmoKPDwvRm9
udCA3IDAgUiA+PgplbmRvYmoKNCAwIG9iago8PC9GaWx0ZXIgL0ZsYXRlRGVjb2RlIC
9MZW5ndGggNjAgPj4Kc3RyZWFtCnic0zdUMDZSCEnjcgrhMlQwAEKwgLm5sZ6FuZGBo
UJILpeGR2pOTr5CeH5RToqipkJIFpdrCBcAWMsNBgplbmRzdHJlYW0KZW5kb2JqCjUg
MCBvYmoKPDwvUGFyZW50IDIgMCBSIC9UeXBlIC9QYWdlIC9Db250ZW50cyA0IDAgUiA
+PgplbmRvYmoKNiAwIG9iago8PC9CYXNlRm9udCAvQ291cmllci1Cb2xkIC9TdWJ0eX
BlIC9UeXBlMSAvRW5jb2RpbmcgL1dpbkFuc2lFbmNvZGluZyAvVHlwZSAvRm9udCA+P
```

```
gplbmRvYmoKNyAwIG9iago8PC8xIDYgMCBSID4+CmVuZG9iago4IDAgb2JqCjw8L0Rp
c3BsYXlEb2NUaXRsZSB0cnVlID4+CmVuZG9iago5IDAgb2JqCjw8L1N1YmplY3QgKP7
/AFMAYQBtAHAAbABlACAAYQBiAG8AdQB0ACAAYQAgAHMAaQBtAHAAbABlACAAJwBoAG
UAbABsAG8AIAB3AG8AcgBsAGQAJwAgAHUAcwBpAG4AZwAgAFAARABGACAAQwBsAG8Ad
wBuKSAvQ3JlYXRvciAo/v8AbwByAGcALgBwAGQAZgBjAGwAbwB3AG4ALgBzAGEAbQBw
AGwAZQBzAC4AYwBsAGkALgBIAGUAbABsAG8AVwBvAHIAbABkAFMAYQBtAHAAbABlKSA
vQXV0aG9yICj+/wBTAHQAZQBmAGEAbgBvACAAQwBoAGkAegB6AG8AbABpAG4AaSkgL0
NyZWF0aW9uRGF0ZSAoRDoyMDExMDMwNTIwMDYxNSswMScwMCcpIC9Qcm9kdWNlciAo/
v8AUABEAEYAIABDAGwAbwB3AG4AIABmAG8AcgAgAEoAYQB2AGEAIAAwAC4AMQAuADAp
IC9UaXRsZSAo/v8AUABEAEYAIABDAGwAbwB3AG4AIAAtACAASABlAGwAbABvACAAdwB
vAHIAbABkACAAcwBhAG0AcABsAGUpID4+CmVuZG9iagp4cmVmCjAgMTAKMDAwMDAwMD
AwMCA2NTUzNSBmDQowMDAwMDAwMDE1IDAwMDAwIG4NCjAwMDAwMDAwODggMDAwMDAgb
g0KMDAwMDAwMDE4NyAwMDAwMCBuDQowMDAwMDAwMjE5IDAwMDAwIG4NCjAwMDAwMDAz
NDkgMDAwMDAgbg0KMDAwMDAwMDQxMSAwMDAwMCBuDQowMDAwMDAwNTEwIDAwMDAwIG4
NCjAwMDAwMDA1MzkgMDAwMDAgbg0KMDAwMDAwMDU4MSAwMDAwMCBuDQp0cmFpbGVyCj
w8L1Jvb3QgMSAwIFIgL0luZm8gOSAwIFIgL1NpemUgMTAgPj4Kc3RhcnR4cmVmCjEwM
zcKJSVFT0Y="
            }
          ]
        },
        {
          "fieldValues": [
            "3",
            "mdi.5678@foobarcorp.com"
          ],
          "optionalData": [
            {
              "name": "FIRST_NAME",
              "value": "John"
            },
            {
              "name": "LAST_NAME",
              "value": "Doe"
            }
          ],
          "attachmentData": [
            {
              "name": "Bye.asd",
              "value":
"JVBERi0xLjQKJYCAgIAKMSAwIG9iago8PC9QYWdlcyAyIDAgUiAvVmlld2VyUHJlZm
VyZW5jZXMgOCAwIFIgL1R5cGUgL0NhdGFsb2cgPj4KZW5kb2JqCjIgMCBvYmoKPDwvQ
291bnQgMSAvTWVkaWFCb3ggWzAgMCA1OTYgODQyIF0gL1R5cGUgL1BhZ2VzIC9SZXNv
dXJjZXMgMyAwIFIgL0tpZHMgWzUgMCBSIF0gPj4KZW5kb2JqCjMgMCBvYmoKPDwvRm9
udCA3IDAgUiA+PgplbmRvYmoKNCAwIG9iago8PC9GaWx0ZXIgL0ZsYXRlRGVjb2RlIC
9MZW5ndGggNjAgPj4Kc3RyZWFtCnic0zdUMDZSCEnjcgrhMlQwAEKwgLm5sZ6FuZGBo
UJILpeGR2pOTr5CeH5RToqipkJIFpdrCBcAWMsNBgplbmRzdHJlYW0KZW5kb2JqCjUg
```

```
MCBvYmoKPDwvUGFyZW50IDIgMCBSIC9UeXBlIC9QYWdlIC9Db250ZW50cyA0IDAgUiA
+PgplbmRvYmoKNiAwIG9iago8PC9CYXNlRm9udCAvQ291cmllci1Cb2xkIC9TdWJ0eX
BlIC9UeXBlMSAvRW5jb2RpbmcgL1dpbkFuc2lFbmNvZGluZyAvVHlwZSAvRm9udCA+P
gplbmRvYmoKNyAwIG9iago8PC8xIDYgMCBSID4+CmVuZG9iago4IDAgb2JqCjw8L0Rp
c3BsYXlEb2NUaXRsZSB0cnVlID4+CmVuZG9iago5IDAgb2JqCjw8L1N1YmplY3QgKP7
/AFMAYQBtAHAAbABlACAAYQBiAG8AdQB0ACAAYQAgAHMAaQBtAHAAbABlACAAJwBoAG
UAbABsAG8AIAB3AG8AcgBsAGQAJwAgAHUAcwBpAG4AZwAgAFAARABGACAAQwBsAG8Ad
wBuKSAvQ3JlYXRvciAo/v8AbwByAGcALgBwAGQAZgBjAGwAbwB3AG4ALgBzAGEAbQBw
AGwAZQBzAC4AYwBsAGkALgBIAGUAbABsAG8AVwBvAHIAbABkAFMAYQBtAHAAbABlKSA
vQXV0aG9yICj+/wBTAHQAZQBmAGEAbgBvACAAQwBoAGkAegB6AG8AbABpAG4AaSkgL0
NyZWF0aW9uRGF0ZSAoRDoyMDExMDMwNTIwMDYxNSswMScwMCcpIC9Qcm9kdWNlciAo/
v8AUABEAEYAIABDAGwAbwB3AG4AIABmAG8AcgAgAEoAYQB2AGEAIAAwAC4AMQAuADAp
IC9UaXRsZSAo/v8AUABEAEYAIABDAGwAbwB3AG4AIAAtACAASABlAGwAbABvACAAdwB
vAHIAbABkACAAcwBhAG0AcABsAGUpID4+CmVuZG9iagp4cmVmCjAgMTAKMDAwMDAwMD
AwMCA2NTUzNSBmDQowMDAwMDAwMDE1IDAwMDAwIG4NCjAwMDAwMDAwODggMDAwMDAgb
g0KMDAwMDAwMDE4NyAwMDAwMCBuDQowMDAwMDAwMjE5IDAwMDAwIG4NCjAwMDAwMDAz
NDkgMDAwMDAgbg0KMDAwMDAwMDQxMSAwMDAwMCBuDQowMDAwMDAwNTEwIDAwMDAwIG4
NCjAwMDAwMDA1MzkgMDAwMDAgbg0KMDAwMDAwMDU4MSAwMDAwMCBuDQp0cmFpbGVyCj
w8L1Jvb3QgMSAwIFIgL0luZm8gOSAwIFIgL1NpemUgMTAgPj4Kc3RhcnR4cmVmCjEwM
zcKJSVFT0Y="
            }
          ]
        }
      ],
      "fieldNames": [
        "CUSTOMER_ID_",
        "EMAIL_ADDRESS_"
      ]
    },
    "mergeRule": {
      "htmlValue": "H",
      "matchColumnName1": "EMAIL_ADDRESS_",
      "matchColumnName2": null,
      "optoutValue": "O",
      "optinValue": "I",
      "insertOnNoMatch": true,
      "defaultPermissionStatus": "OPTIN",
      "rejectRecordIfChannelEmpty": "E",
      "updateOnMatch": "REPLACE_ALL",
      "textValue": "T",
      "matchOperator": "NONE"
    }
  }
```

Response Notes

- A successful response will have an HTTPS code of 200 OK, and the response payload indicates the success or failure for each recipient.
- The following `errorMessage` values are specific to email attachment validations for individual recipients. In each case, `success : false` for the recipient and the email is not sent.
  - FAILURE: Number of attachments cannot exceed 10 for a recipient

    (10-attachment limit includes both API payload and attachments set for the email campaign by the marketer.)
  - FAILURE: Invalid attachment type, allowed types are png, jpg, jpeg, pdf, ical

    (System checks both file name and value. In the example above, bye.asd decodes to a PDF file, but because the file extension is incorrect in the file name, this attachment still causes a failure for the recipient.)
  - FAILURE: The total size of the attachments cannot exceed 500 KB

    (500 KB size limit for attachments includes both API payload and attachments set for the email campaign by the marketer.)
- If the system detects a virus in one of the attachments, the entire request fails with an HTTPS code 400 Bad Request and the errorCode VIRUS_FOUND. The detail returned in the response payload contains the name of the file containing the virus.

Sample Response:

The response returns a status for each recipient record sent in the request payload. In this example:

- The first record was successfully merged and the email was sent to the recipient whose RIID is 72067.

- The second record was successfully merged, but the email was undeliverable because the recipient's email deliverability status in his Profile List record was set to "Undeliverable" in Responsys.
- The third record was successfully merged, but the email was not sent because the attachment had an incorrect file type (Bye.asd).

```
[
 {
  "errorMessage" : null,
  "success" : true,
  "recipientId" : 72067
 },
       {
  "errorMessage" :" RECIPIENT_STATUS_UNDELIVERABLE: Recipient
deliverability status is undeliverable",
  "success" : false,
  "recipientId" : -1
 },
       {
  "errorMessage" :" FAILURE: Invalid attachment type, allowed types
are png, jpg, jpeg, pdf, ical",
  "success" : false,
  "recipientId" : 73159
 }
]
```

Trigger email message

Use this interface to trigger email messages to existing members of a profile list.

REQUEST NOTES:

- You can send Responsys Email campaigns that already exist to up to 200 members of a profile list.
- Do not use system column names for the optional data, such as `EMAIL_ADDRESS_`. This may cause incorrect data to be processed, possibly resulting in skipped records and incorrect data being recorded in the Event table.

- You can also use the Trigger Email Message interface to trigger a proof launch. If you use Trigger Email Message for proof testing, any recipients you include must belong to the profile list associated with the campaign, even if the campaign has a proof/seed list attached. If you try to invoke Trigger Email Message for a recipient from the proof/seed list, the system returns a `RECIPIENT NOT FOUND` error. Note that when the content is changed for the campaign, then a triggered launch will pick up the updated content within 10 minutes. Alternatively, you can use a scheduled launch to see the changes without the delay, and you can also use a scheduled launch if you want to proof test with your proof/seed list. For more details, see Schedule an Email or Push Campaign Launch.
- To pass extended/accented characters in `optionalData` payload, they must be escaped as Unicode characters. For example, the euro symbol € is escaped as `\u20AC`, the yen symbol ¥ is escaped as `\u00A5`, an ü is escaped as `\u00FC`, an é is escaped as `\u00E9`, and the like. Otherwise, you may receive an `INVALID_REQUEST_CONTENT` error.

Service URL:

`/rest/api/v1.3/campaigns/{campaignName}/email`

Required Path Parameter:

`campaignName`

Request Method:

`POST`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Sample Request 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"
          }
     ]
    }
   ]
  }
```

Sample Response:

The response returns a status for each record sent in the request. In this example, email was sent successfully to the recipient whose RIID is 72067. For the second record, the trigger email action was not successful, because the recipient's deliverability status was undeliverable.

```
        [{
         "errorMessage" : null,
         "success" : true,
         "recipientId" : 72067
        }, {
         "errorMessage" :" RECIPIENT_STATUS_UNDELIVERABLE: Recipient
deliverability status is undeliverable ",
         "success" : false,
         "recipientId" : -1
        }
       ]
```

Triggering SMS Messages

Responsys can send personalized SMS messages to mobile phone numbers.

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

Responsys can send personalized SMS messages to mobile phone numbers.

REQUEST NOTES:

- You can send Responsys Email campaigns that already exist to up to 200 members of a profile list.
- To pass extended/accented characters in `optionalData` payload, they must be escaped as Unicode characters. For example, the euro symbol € is escaped as `\u20AC`, the yen symbol ¥ is escaped as `\u00A5`, an ü is escaped as `\u00FC`, an é is escaped as `\u00E9`, and the like. Otherwise, you may receive an `INVALID_REQUEST_CONTENT` error.

Service URL:

`/rest/api/v1.3/campaigns/{campaignName}/email`

Required Path Parameter:

`campaignName`

Request Method:

`POST`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Sample Request 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"
        }
      ]
    }
  ]
}
```

Sample Response:

The response returns a status for each record sent in the request. In this example, email was sent successfully to the recipient whose RIID is 72067. For the second record, the trigger email action was not successful, because the recipient's deliverability status was undeliverable.

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

Triggering Mobile Push Messages

Responsys can deliver personalized messages to mobile apps using Push campaign messaging.

Trigger Push Messages

Use the following interface to trigger Push campaign messages to existing members of a Profile List and its corresponding App Channel List.

REQUEST NOTES:

- You can send push messages to existing Responsys Push campaigns for up to 200 members of a Profile List and its App Channel List.
- The request payload allows specifying one of the available Profile List attributes, so that you may uniquely identify the recipients of the push message.
- The Profile List attributes that can be specified are one of the following: `recipientId`, `customerId`, `emailAddress`, `mobileNumber`, `emailSHA256Hash` or `emailMD5Hash`.

  > **Note**: In the request body, all `recipientData` must be present but the profile list attributes you are not using must be set to `null`.

- Use the `listType` attribute `PROFILE` when recipients need to be matched from Profile List (that is, known recipients).

NOTES:

- The Profile List members selected using these attributes are matched against the App Channel List to find the device IDs of all devices to trigger push messages.
- You can send to all devices and apps that a mobile app user has installed by setting "`apiKey`": `null`.
- You can send to a specific mobile app that a user has installed by specifying the apiKey value. Note that you may only use one `apiKey` value per recipient. This means that if a mobile app end user (recipient) has a single entry in a Profile List but has multiple entries in the App Channel List for different apps, the apiKey attribute restricts the push message to trigger for one of those apps.

- Use the `listType` attribute `PUSH` when recipients need to be matched from App Channel List (that is, unknown recipients).

  NOTES:

  - In the request body, all `recipientData` properties must be present but the Profile List attributes you are not using must be set to `null`.
  - When using `listType` of `PUSH`, the App Channel List attributes that you must specify are **both** `deviceId` and `apiKey`. Including both `apiKey` and `deviceId` ensures that push messages will only be triggered for devices that match the `apiKey`.
  - You can use the `optionalData` attribute as part of the payload to customize the push message.
    - Do not use system-defined fields (for example, `MOBILE_NUMBER_`) for your optional data.
    - To pass extended/accented characters in `optionalData` payload, they must be escaped as Unicode characters. For example, the euro symbol € is escaped as `\u20AC`, the yen symbol ¥ is escaped as `\u00A5`, an ü is escaped as `\u00FC`, an é is escaped as `\u00E9`, and the like. Otherwise, you may receive an `INVALID_REQUEST_CONTENT` error.

Service URL:

```
/rest/api/v1.3/campaigns/{campaignName}/push
```

Required Path Parameter:

`campaignName`

Request Method:

`POST`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Sample Request Body:

The following Trigger Push Message request payload example is requesting that Responsys send a Push campaign message to two recipients:

- The first recipient is an unknown user. Unknown users are those for whom Responsys has device data in the App Channel List, but there is no match between the App Channel List record and a Profile List record. Because the user is not known, the recipient's `listType` is `PUSH`, `deviceId` contains recipient device's `deviceId`, and `apiKey` contains the `API_KEY` value of the mobile app installed on the recipient device.
- The second recipient is a known user. Known users are those for whom Responsys has a matching Profile List record for an App Channel List record. Because the user is known, the recipient's `listType` is `PROFILE`. Also, because the `API_KEY` for the mobile app specified in the `apiKey` property, the push message is being sent to that specific app. When the apiKey property is `null`, Responsys sends the push message to all device + app combinations that the known mobile app user has installed.

```
{
  "recipientData": [
    {
      "customerId": null,
      "emailAddress": null,
      "recipientId": null,
      "mobileNumber": null,
      "emailSHA256Hash": null,
      "emailMD5Hash": null,
      "deviceId": "<device Id value>",
      "apiKey": "<API Key value>",
      "listType": "PUSH",
      "optionalData": [
        {
          "name": "CUSTOM1",
          "value": "c1a_value_new"
        },
        {
          "name": "CUSTOM2",
          "value": "c2a_value_new"
        }
      ]
    },
    {
      "customerId": null,
      "emailAddress": "foo.bar@oracle.com",
      "recipientId": null,
      "mobileNumber": null,
      "emailSHA256Hash": null,
      "emailMD5Hash": null,
      "deviceId": null,
      "apiKey": "<API Key value>",
      "listType": "PROFILE",
      "optionalData": [
        {
          "name": "CUSTOM1",
          "value": "c1a_value_new"
        },
        {
          "name": "CUSTOM2",
          "value": "c2a_value_new"
        }
      ]
    }
  ]
}
```

RESPONSE NOTES:

- The `recipientId` displayed in the Trigger Push Message response body is the App Channel List RIID. (This differs from the API responses for sending to other channels, such as

Email and SMS, which use the RIID from the Profile List.)

- The `success` results returned in the Trigger Push Message response body may vary from actual results, depending on the intended recipient's `CHANNEL_DELIVERABILITY_STATUS_` and `CHANNEL_PERMISSION_STATUS_` settings in the App Channel List. Consult the EventDB to verify whether the system sent the messages.
    - When `CHANNEL_DELIVERABILITY_STATUS_` is `D` (Deliverable) but `CHANNEL_PERMISSION_STATUS_` is `O` (OptOut):

      If the request was sent with `listType` of `PUSH`, the API response returns `"success":true` but in the EventDB, the record is shown as SKIPPED because the recipient is not found. This occurs whether a record is present in the Profile List or not.

      However, when request is sent with `listType` of `PROFILE` and a record *is* present in the Profile List, the API response sends `"success":false` and `"errorMessage" : "RECIPIENT_STATUS_UNDELIVERABLE: Recipient deliverability status is undeliverable"`.
    - When `CHANNEL_DELIVERABILITY_STATUS_` is `U` (Undeliverable) and `CHANNEL_PERMISSION_STATUS_` is `O` (OptOut), the API response returns `"success":true` but in the EventDB, the record is shown as SKIPPED because the recipient is not found. This occurs regardless of the `listType` setting and whether or not there is a record present in the Profile List.

Sample Response in case of success:

The response returns information about each record sent in the request. In this example, the mobile push message was sent successfully to the recipient device whose App Channel List RIID is 72067. For the second record, the trigger push message action was not successful, because `listType` was set to `PROFILE`, a record was present in the Profile List, `CHANNEL_DELIVERABILITY_STATUS_` is `D` (Deliverable) in the App Channel List, but the recipient's `CHANNEL_PERMISSION_STATUS_` in the App Channel List was set to "`O`" (OptOut).

```
[
  {
    "errorMessage": null,
    "success": true,
    "recipientId": 72067
  },
  {
    "errorMessage": "RECIPIENT_STATUS_UNDELIVERABLE: Recipient
deliverability status is undeliverable",
    "success": false,
    "recipientId": -1
  }
]
```

Sample Response in case of failure:

In this example, none of the recipients received the mobile push message, because the system could not find the campaign name sent in the request.

```
{
   "type": "",
   "title": "Campaign not found",
   "errorCode": "CAMPAIGN_NOT_FOUND",
   "detail": "Campaign not found with name [campaignName]" ,
   "errorDetails": []
}
```

Organizations

Fetch all organizations

Fetch a program's organization hierarchy

Get a campaign's organization hierarchy

Update a program's organizational access

Update a campaign's organizational access

Fetch all organizations

Fetch a list of all organizations within the account.

Service URL:

```
/rest/api/v1.3/attributes/organizations
```

Request Method:

```
GET
```

Request Header:

```
Authorization=<AUTH_TOKEN>

Content-Type=application/json
```

Sample Request URL:

```
/rest/api/v1.3/attributes/organizations
```

Sample Response in case of success:

```
{
  "attributes": {
    "organizations": {
      "attributeValues": [
        {
          "id": 7,
          "name": "Root",
          "lineage": "7:",
          "children": [
            {
              "id": 4000007,
              "name": "India",
              "lineage": "7:4000007:",
              "children": [
                {
                  "id": 4000027,
                  "name": "Karnataka",
                  "lineage": "7:4000007:4000027:",
                  "children": [
                    {
                      "id": 4000067,
                      "name": "Bangalore",
                      "lineage": "7:4000007:4000027:4000067:"
                    },
                    {
                      "id": 4000087,
                      "name": "Mysore",
                      "lineage": "7:4000007:4000027:4000087:"
                    }
                  ]
                },
                {
                  "id": 4000047,
                  "name": "Delhi",
                  "lineage": "7:4000007:4000047:"
                },
                {
                  "id": 8000107,
                  "name": "Tamil Nadu",
                  "lineage": "7:4000007:8000107:",
                  "children": [
                    {
                      "id": 8000127,
                      "name": "Chennai",
                      "lineage": "7:4000007:8000107:8000127:"
                    }
                  ]
                }
              ]
            },
            {
              "id": 27,
              "name": "Legacy",
              "lineage": "7:27:"
            },
            {
              "id": 8000007,
              "name": "US",
              "lineage": "7:8000007:",
              "children": [
                {
                  "id": 8000027,
                  "name": "New York",
                  "lineage": "7:8000007:8000027:",
                  "children": [
                    {
```

```
                    "id": 8000067,
                    "name": "Albany",
                    "lineage": "7:8000007:8000027:8000067:"
                  }
                ]
              },
              {
                "id": 8000047,
                "name": "New Jersey",
                "lineage": "7:8000007:8000047:",
                "children": [
                  {
                    "id": 8000087,
                    "name": "Trenton",
                    "lineage": "7:8000007:8000047:8000087:"
                  }
                ]
              }
            ]
          }
        ]
      }
    ]
  }
 }
}
```

Fetch a program's organization hierarchy

Fetch a list of organizations that can access a program.

Service URL:

`/rest/api/v1.3/attributes/program/{programName}`

Required Path Parameter:

`programName`

Request Method:

`GET`

Request Header:

```
Authorization=<AUTH_TOKEN>

Content-Type=application/json
```

Sample Request URL:

```
/rest/api/v1.3/attributes/programs/example_program
```

Sample Response in case of success:

```
{
  "attributes": {
    "organizations": {
      "attributeValues": [
        {
          "id": 7,
          "name": "USA",
          "lineage": "7:"
        }
      ]
    }
  }
}
```

Sample Response in case of failure:

If the campaign does not exist, the error will resemble the following:

```
{
  "type": "",
  "title": "Program not found",
  "errorCode": "PROGRAM_NOT_FOUND",
  "detail": "Program with the name example_progrm not found",
  "errorDetails": []
}
```

Get a campaign's organization hierarchy

Fetch a list of organizations that can access a campaign.

Service URL:

`/rest/api/v1.3/attributes/campaigns/{campaignName}`

Required Path Parameter:

`campaignName`

Request Method:

`GET`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Sample Request URL:

```
/rest/api/v1.3/attributes/campaigns/example_campaign
```

Sample Response in case of success:

```
{
  "attributes": {
    "organizations": {
      "attributeValues": [
        {
          "id": 7,
          "name": "USA",
          "lineage": "7:"
        }{
          "id": 8,
          "name": "Asia",
          "lineage": "7:10005"
        }
      ]
    }
  }
}
```

Sample Response in case of failure:

If the campaign does not exist, the error will resemble the following:

```
{
    "type": "",
    "title": "Campaign not found",
    "errorCode": "CAMPAIGN_NOT_FOUND",
    "detail": "Campaign with the name example_campaig not found",
    "errorDetails": []
}
```

Update a program's organizational access

Update the organizations that can access a program.

Service URL:

`/rest/api/v1.3/attributes/campaigns/{campaignName}`

Required Path Parameter:

`programName`

Request Method:

`POST`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Sample Request Body:

```
{
  "attributes": {
    "organizations": {
      "attributeValues": [
        {
          "name": "USA"
        },
        {
          "name": "APAC"
        },
        {
          "name": "Global"
        }
      ]
    }
  }
}
```

Sample Response in case of success:

```
{
    "attributes": {
        "organizations": {
            "attributeValues": [
                {
                    "id": 7,
                    "name": "USA",
                    "lineage": "7:"
                },
                {
                    "id": 4000007,
                    "name": "APAC",
                    "lineage": "7:4000007:"
                },
                {
                    "id": 12000007,
                    "name": "Global",
                    "lineage": "7:12000007:"
                }
            ]
        }
    }
}
```

Sample Response in case of failure:

If the attribute does not match an existing organization, the following error will be returned.

```
{
  "type": "",
  "title": "Set Object Attribute Exception",
  "errorCode": "SET_OBJECT_ATTRIBUTE_EXCEPTION",
  "detail": "Unable to set Access Attributes for the program.
Invalid attribute value: Usa",
  "errorDetails": []
}
```

Update a campaign's organizational access

Update the organizations that can access a campaign.

Service URL:

`/rest/api/v1.3/attributes/campaigns/{campaignName}`

Required Path Parameter:

`campaignName`

Request Method:

`POST`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Sample Request Body:

```
{
  "attributes": {
    "organizations": {
      "attributeValues": [
        {
          "name": "USA"
        },
        {
          "name": "APAC"
        },
        {
          "name": "Global"
        }
      ]
    }
  }
}
```

Sample Response in case of success:

```
{
    "attributes": {
        "organizations": {
            "attributeValues": [
                {
                    "id": 7,
                    "name": "USA",
                    "lineage": "7:"
                },
                {
                    "id": 4000007,
                    "name": "APAC",
                    "lineage": "7:4000007:"
                },
                {
                    "id": 12000007,
                    "name": "Global",
                    "lineage": "7:12000007:"
                }
            ]
        }
    }
}
```

Sample Response in case of failure:

If the attribute does not match an existing organization, the following error will be returned.

```
{
  "type": "",
  "title": "Set Object Attribute Exception",
  "errorCode": "SET_OBJECT_ATTRIBUTE_EXCEPTION",
  "detail": "Unable to set Access Attributes for the campaign.
Invalid attribute value: Usa",
  "errorDetails": []
}
```

Raising Custom Events for Cross-channel Marketing Programs

Responsys users can set up a Responsys Program orchestration to listen for one or more custom events. Custom events can start a Program orchestration or can be used in a Program orchestration event switch.

Get all custom events for an account

Use this interface to retrieve the names and descriptions of all custom events for an account.

Service URL:

`/rest/api/v1.3/events`

Request Method:

`GET`

Request Header:

`Authorization=<AUTH_TOKEN>`

Request Body:

`None`

Sample Response in case of success:

RESPONSE NOTE: The response is a list of custom event names and their descriptions.

```
[
  {
    "eventName": "My_Test",
    "description": "This is a test custom event.",
    "eventType": "1"
  },
  {
    "eventName": "My_Test_Mobile",
    "description": "Test custom event that includes mobile RIIDs",
    "eventType": "1"
  },
  {
    "eventName": "Test3",
    "description": "Test event 3",
    "eventType": "1"
  }
]
```

Sample Response in case of failure:

```
{
  "type": "",
  "title": "Custom event not found",
  "errorCode": "CUSTOM_EVENT_NOT_FOUND",
  "detail": "Account : acctname : does not have any custom
events.",
  "errorDetails": []
}
```

Trigger a custom event

Use the following API to trigger a specific custom event. The Responsys Program orchestration will use the existing members of a profile list that are specified in the API request.

NOTES:

- A single request is limited to 200 recipients. If you need to trigger a custom event for more than 200 recipients, then you should submit multiple requests to trigger the custom event.
- Responsys requires Enactment Batching feature to be enabled for the account when using trigger custom event with mobile app campaigns in Program. Otherwise, the mobile app campaign events in the program will not be processed. However, there are some tradeoffs to

consider before enabling the feature. Please refer to How Enactment Batching Affects Processing for more details.

If you have the Real-time Events feature enabled for your account, and you need to send near-real-time messages based on the custom event, then ensure that the custom event you specify is of type Real-time. The Get Custom Events API does not return custom event types in its response, but you can view the custom events and their types by accessing the Define Custom Event Types page in the Account administrator section of Responsys. Contact the Responsys Account Administrator for assistance if you do not have access. For more information, see the Defining Custom Event Types topic in the *Oracle Responsys Help Center*.

When Enactment Batching is used, the Responsys Account Admin must check the “Include Mobile App Channel RIIDs” check box for the custom event’s definition. (In Responsys, this setting is found in Accounts > Account Customization > Global Settings > Define custom event types). With this option set, Responsys creates an enactment for each device associated with every Profile RIID; otherwise, the Push channel enactments will NOT enter the program and by extension, it will not send any push messages to the intended recipients. To avoid duplicate emails when email is part of the orchestration, marketers who orchestration the program must add a data switch immediately before the Email event.

- The Profile List parameters that can be specified are one of recipientId, emailAddress, customerId, or mobileNumber. The system processes the attributes in the above-listed order and accepts the first non-null value found.
- To pass extended/accented characters in optionalData payload, they must be escaped as Unicode characters. For example, the euro symbol € is escaped as \u20AC, the yen symbol ¥ is escaped as \u00A5, an ü is escaped as \u00FC, an é is escaped as \u00E9, and the like. Otherwise, you may receive an INVALID_REQUEST_CONTENT error.
- Sending duplicate names in the recipientData results in an error message (MULTIPLE_RECIPIENTS_FOUND).

Service URL:

`/rest/api/v1.3/events/{eventName}`

Required Path Parameter:

`eventName` - Name of the custom event defined in Responsys.

Request Method:

`POST`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Sample Request Body:

```
{
  "customEvent": {
    "eventNumberDataMapping": null,
    "eventDateDataMapping": null,
    "eventStringDataMapping": null
  },
  "recipientData": [
    {
      "recipient": {
        "customerId": 1,
        "emailAddress": null,
        "listName": {
          "folderName": "WS_REST_SAMPLE",
          "objectName": "wsrest"
        },
        "recipientId": null,
        "mobileNumber": null,
        "emailFormat": "HTML_FORMAT"
      },
      "optionalData": [
        {
          "name": "CUSTOM1",
          "value": "value1"
        }
      ]
    }
  ]
}
```

Sample Response:

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

Managing Campaign Launch Schedules (Email or Push)

The following interfaces can be used to manage existing Responsys Email campaign or Push campaign launch schedule objects.

Notes:

- In the sections that follow, content applies to both Email and Push, unless specifically noted.
- A Responsys user must create the campaign in Responsys, and the campaign must not have validation errors.
- For Email campaigns, the interfaces apply for only Email Message Designer (EMD) campaigns; the interfaces do not support classic campaigns.
- For Push campaigns, your Push campaign must have the “From address” set in the Launch Options of the campaign workbook. If the “From address” is not set, the campaign will be scheduled successfully, but the campaign *launch* will fail.
- The APIs support Push campaigns only for the Mobile App channel. These APIs do not support In-app Message campaigns.

Schedule an Email or Push Campaign Launch

Use this interface to create a campaign launch schedule for an existing Email or Push campaign. You can schedule the campaign for immediate or future launch. You can also schedule launches for proof testing an existing Email campaign.

Service URL:

```
/rest/api/v1.3/campaigns/{campaignName}/schedule
```

Required Path Parameter:

`campaignName`

Request Method:

```
POST
```

Request Header:

```
Authorization=<AUTH_TOKEN>

Content-Type=application/json
```

Required Request Body Parameters:

Minimum required attributes for Email and all required for Push:

- `scheduleType` (ONCE or NOW)
- `scheduledTime` (Use when scheduleType=ONCE. Date must be in the format YYYY-MM-DD HH:MM xx, where xx is AM or PM)
- `launchOptions`:
    - `progressEmailAddresses` (one or more email addresses)
    - `progressChunk` (CHUNK_10K, CHUNK_50K, CHUNK_100K, CHUNK_500K, or CHUNK_1M)

Attributes for Email proof launch:

- `scheduleType` (ONCE or NOW)
- `scheduledTime` (Use when scheduleType=ONCE. Date must be in the format YYYY-MM-DD HH:MM xx, where xx is AM or PM)
- `launchOptions`:
    - `proofLaunch` (true if a proof launch should be performed; otherwise false or omit)
    - `proofLaunchEmail` (Email address for the proof launch.)
    - `proofLaunchType` (LAUNCH_TO_ADDRESS or LAUNCH_TO_PROOFLIST or LAUNCH_TO_ADDRESS_USING_PROOFLIST)
    - `recipientLimit` (integer; recipient limit)
    - `samplingNthSelection` (integer; sampling selection)
    - `samplingNthOffset` (integer; sampling offset)
    - `samplingNthInterval` (integer; sampling interval)
    - `progressEmailAddresses` (one or more email addresses)
    - `progressChunk` (CHUNK_10K, CHUNK_50K, CHUNK_100K, CHUNK_500K, or CHUNK_1M)

Sample Requests and Responses

The following sections, Proof Email Examples, Email Examples, and Push Examples, show the sample request and response bodies for the different types of campaign scheduling.

NOTES:

- A success response returns the launch ID number, the scheduling attributes sent in the request, and the campaign-specific links for related API calls.
- Error responses occur when:

- Scheduled date and time are in the past
- Campaign is already scheduled for launch at the requested date and time
- Invalid campaign
- Invalid parameters
- API user has insufficient access to launch a campaign

Proof Test Email Examples

Sample Request Body – Email proof launch to an address

In the following example request, the schedule launch request is for a single proof launch at 6:00 AM on May 25, 2019. The proof launch is to a single address, someemail@a.com. Launch progress emails will be sent to email1@a.com and email2@a.com.

```
{
  "scheduleType": "ONCE",
  "scheduledTime": "2019-05-25 06:00 AM",
  "launchOptions": {
    "proofLaunch": true,
    "proofLaunchEmail": "someemail@a.com",
    "proofLaunchType": "LAUNCH_TO_ADDRESS",
    "recipientLimit": 3,
    "samplingNthSelection": 1,
    "samplingNthOffset": 1,
    "samplingNthInterval": 1,
    "progressEmailAddresses": [
      "email1@a.com",
      "email2@a.com"
    ],
    "progressChunk": "CHUNK_10K"
  }
}
```

Sample Request Body – Email proof launch to a proof list

You can also sent a proof launch to the proof list. In the following example, the proof test is scheduled for an immediate launch, and it will be sent to two recipients in the proof list associated to the campaign. You can set the number of recipients by using the `recipientLimit` attribute.

```
{
    "scheduleType": "NOW",
    "launchOptions": {
      "proofLaunch": true,
      "proofLaunchType" : "LAUNCH_TO_PROOFLIST",
      "recipientLimit": 2
    }
}
```

Sample Response Body – Email proof launch

In this example, the `POST` request endpoint was as follows, where **MyTestCampaign** is the campaign name:

`/rest/api/v1.3/campaigns/MyTestCampaign/schedule`

The response returns a launch id (1), which you can use in other Launch Schedule requests (as shown in the `links` array), and it echoes back the attributes sent in the request. In this case, the response is for an email proof launch sent to an address.

```
{
    "id": 1,
    "scheduleType": "ONCE",
    "scheduledTime": "2019-05-25 06:00 AM",
    "launchOptions": {
        "proofLaunch": true,
        "proofLaunchEmail": "someemail@a.com",
        "proofLaunchType": "LAUNCH_TO_ADDRESS,
        "recipientLimit": 3,
        "samplingNthSelection": 1,
        "samplingNthOffset": 1,
        "samplingNthInterval": 1,
        "progressEmailAddresses": [
            "email1@a.com",
            "email2@a.com"
        ],
        "progressChunk": "CHUNK_10K",
        "links": [
            {
                "rel": "self",
                "href":
"/rest/api/v1.3/campaigns/MyTestCampaign/schedule",
                "method": "POST"
            },
            {
                "rel": "getSchedule",
                "href":
"/rest/api/v1.3/campaigns/MyTestCampaign/schedule/1",
                "method": "GET"
            },
            {
                "rel": "updateSchedule",
                "href":
"rest/api/v1.3/campaigns/MyTestCampaign/schedule/1",
                "method": "PUT"
            },
            {
                "rel": "deleteSchedule",
                "href":
"rest/api/v1.3/campaigns/MyTestCampaign/schedule/1",
                "method": "DELETE"
            }
        ]
    }
```

Email Examples

Sample Request Body - Email

In this example, we are scheduling a campaign launch for December 30, 2019 at 11:17 AM of an EMD email campaign named **JMP emailTest for API**. Because `proofLaunch` is set to `false`, this is a campaign launch scheduled for the targeted recipients of the email campaign and not a proof launch.

```
POST /rest/api/v1.3/campaigns/JMP emailTest for API/schedule
```

```
{
  "scheduleType": "ONCE",
  "scheduledTime": "2019-12-30 11:17 AM",
  "launchOptions": {
    "proofLaunch": false,
    "progressEmailAddresses": [
    "email1@ora.com",
    "email2@ora.com"],
    "progressChunk": "CHUNK_10K"
  }
}
```

Sample Response Body - Email

The response returns a launch `id` (244798), which you can use in other Launch Schedule requests (as shown in the `links` array), and it echoes back the attributes sent in the request. In this case, the response is for a live email campaign launch (not a proof launch) of **JMP emailTest for API**.

```
{
  "id": 244798,
  "scheduleType": "ONCE",
  "scheduledTime": "2019-12-30 11:17 AM",
  "launchOptions": {
    "proofLaunch": false,
    "progressEmailAddresses": [
    "email1@ora.com",
    "email2@ora.com"
  ],
  "progressChunk": "CHUNK_10K"
  },
  "links": [
    {
      "rel": "self",
      "href": "/rest/api/v1.3/campaigns/JMP emailTest for
API/schedule",
      "method": "POST"
    },
    {
       "rel": "deleteSchedule",
       "href": "/rest/api/v1.3/campaigns/JMP emailTest for
API/schedule/244798",
       "method": "DELETE"
    },
    {
      "rel": "updateSchedule",
      "href": "/rest/api/v1.3/campaigns/JMP emailTest for
API/schedule/244798",
      "method": "PUT"
    },
    {
      "rel": "getSchedule",
      "href": "/rest/api/v1.3/campaigns/JMP emailTest for
API/schedule/244798",
      "method": "GET"
    }
  ]
}
```

Push Examples

Sample Request Body – Push

In this example, we are scheduling a campaign launch for December 30, 2019 at 11:17 AM of a Push campaign named **JMP Test for API**.

```
POST /rest/api/v1.3/campaigns/JMP emailTest for API/schedule

{
  "scheduleType": "ONCE",
  "scheduledTime": "2019-12-30 11:17 AM",
  "launchOptions": {
    "progressEmailAddresses": [
      "email1@ora.com",
      "email2@ora.com"
    ],
    "progressChunk": "CHUNK_10K"
  }
}
```

Sample Response Body – Email

The response returns a launch `id` (244797), which you can use in other Launch Schedule requests (as shown in the `links` array), and it echoes back the attributes sent in the request.

```
{
  "id": 244797,
  "scheduleType": "ONCE",
  "scheduledTime": "2019-12-30 11:17 AM",
  "launchOptions": {
    "proofLaunch": false,
    "progressEmailAddresses": [
      "email1@ora.com",
      "email2@ora.com"
    ],
    "progressChunk": "CHUNK_10K"
  },
  "links": [
    {
      "rel": "self",
      "href": "/rest/api/v1.3/campaigns/JMP Test for API/schedule",
      "method": "POST"
    },
    {
      "rel": "deleteSchedule",
      "href": "/rest/api/v1.3/campaigns/JMP Test for
API/schedule/244797",
      "method": "DELETE"
    },
    {
      "rel": "updateSchedule",
      "href": "/rest/api/v1.3/campaigns/JMP Test for
API/schedule/244797",
      "method": "PUT"
    },
    {
      "rel": "getSchedule",
      "href": "/rest/api/v1.3/campaigns/JMP Test for
API/schedule/244797",
      "method": "GET"
    }
  ]
}
```

Get a Launch Schedule for an Email or Push Campaign

Use this interface to get the schedule of an Email or Push campaign by using the campaign schedule ID that was returned from the schedule campaign API.

Service URL:

`/rest/api/v1.3/campaigns/{campaignName}/schedule/{scheduleId}`

OR

`/rest/api/v1.3/campaigns?type=email`

Required Path Parameters:

- `campaignName`
- `scheduleId` (this can be obtained from the id parameter from the response to either the “Get All Launch Schedules for an Email or Push Campaign” task or the “Schedule an Email or Push Campaign Launch” task)

Request Method:

`GET`

Request Header:

`Authorization=<AUTH_TOKEN>`

Request Body - Required Properties:

`None`

Sample Request Body

Not applicable

Sample Response Body

The following response example was for a proof launch of an Email campaign named **test** with a launch schedule ID of **1**.

```
{
  "id": 1,
  "scheduleType": "ONCE",
  "scheduledTime": "2019-01-25 06:00 AM",
  "launchOptions": {
    "proofLaunch": true,
    "proofLaunchEmail": "someemail@a.com",
    "proofLaunchType": "LAUNCH_TO_ADDRESS",
    "recipientLimit": 3,
    "samplingNthSelection": 1,
    "samplingNthOffset": 1,
    "samplingNthInterval": 1,
    "progressEmailAddresses": [
      "email1@a.com",
      "email2@a.com"
    ],
    "progressChunk": "CHUNK_10K",
    "links": [
      {
        "rel": "self",
        "href": "/rest/api/v1.3/campaigns/test/schedule/1",
        "method": "POST"
      },
      {
        "rel": "createSchedule",
        "href": "/rest/api/v1.3/campaigns/test/schedule",
        "method": "GET"
      },
      {
        "rel": "updateSchedule",
        "href": "rest/api/v1.3/campaigns/test/schedule/1",
        "method": "PUT"
      },
      {
        "rel": "deleteSchedule",
        "href": "rest/api/v1.3/campaigns/test/schedule/1",
        "method": "DELETE"
      }
    ]
  }
}
```

Get All Launch Schedules for an Email or Push Campaign

Use this interface to get all the schedules for an Email or a Push campaign, including all schedule IDs for the launches.

Service URL:

```
/rest/api/v1.3/campaigns/{campaignName}/schedule
```

Required Path Parameter:

`campaignName`

Request Method:

```
GET
```

Request Header:

```
Authorization=<AUTH_TOKEN>
```

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 Body:

```
None
```

Sample Response Body

The following successful response is for a campaign named **testCampaign** that has two active launches. The active launch with Scheduled ID **307357** is of type **RECURRING** that occurs daily, starting on February 9, 2017 and ending on February 23, 2017. The active launch with Schedule ID **307377** is of type **ONCE** that will occur on February 3, 2017 at noon.

```
{
  "schedules": [
    {
      "id": 307357,
      "scheduleType": "RECURRING",
      "scheduledTime": "2017-02-09 10:00 AM",
      "recurringEndTime": "2017-02-23 12:00 AM",
      "recurringInterval": "DAILY",
      "launchOptions": {
        "proofLaunch": false
      },
      "links": [
        {
          "rel": "self",
          "href":
"/rest/api/v1.3/campaigns/testCampaign/schedule/307357",
          "method": "GET"
        },
        {
          "rel": "deleteSchedule",
          "href":
"/rest/api/v1.3/campaigns/testCampaign/schedule/307357",
          "method": "DELETE"
        },
        {
          "rel": "updateSchedule",
          "href":
"/rest/api/v1.3/campaigns/testCampaign/schedule/307357",
          "method": "PUT"
        },
        {
          "rel": "createSchedule",
          "href": "/rest/api/v1.3/campaigns/testCampaign/schedule",
          "method": "POST"
        }
      ]
    },
    {
      "id": 307377,
      "scheduleType": "ONCE",
      "scheduledTime": "2017-02-03 12:00 PM",
      "launchOptions": {
        "proofLaunch": false
      },
      "links": [
        {
          "rel": "self",
          "href":
"/rest/api/v1.3/campaigns/testCampaign/schedule/307377",
          "method": "GET"
        },
        {
          "rel": "deleteSchedule",
          "href":
"/rest/api/v1.3/campaigns/testCampaign/schedule/307377",
          "method": "DELETE"
        },
        {
          "rel": "updateSchedule",
          "href":
"/rest/api/v1.3/campaigns/testCampaign/schedule/307377",
          "method": "PUT"
        },
        {
          "rel": "createSchedule",
          "href": "/rest/api/v1.3/campaigns/testCampaign/schedule",
          "method": "POST"
        }
      ]
    }
  ],
  "links": [
    {
      "rel": "self",
      "href": "/rest/api/v1.3/campaigns/testCampaign/schedule",
      "method": "GET"
    }
  ]
}
```

Sample Response in case of failure:

```
{
  "title": "Campaign not found",
  "errorCode": "CAMPAIGN_NOT_FOUND",
  "detail": "Campaign not found with name [campaignName]"
}
```

Update Email or Push Campaign Launch Schedule

Use this interface to update the schedule of an existing Email or Push campaign, by using the schedule ID that was returned from schedule campaign API.

Service URL:

`/rest/api/v1.3/campaigns/{campaignName}/schedule/{scheduleID}`

Required Path Parameter:

- `campaignName`
- `scheduleId` (this can be obtained from the id parameter from response to either the "Get All Launch Schedules for an Email or Push Campaign" task or the "Schedule an Email or Push Campaign Launch" task)

Request Method:

`PUT`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Required Body Parameters:

- `scheduleType` (ONCE or NOW)
- `scheduledTime` (Date in the format YYYY-MM-DD HH:MM AM or PM)

Sample Request Body:

In this example, we have a campaign launch with schedule ID of **244797** with a scheduled time of **2016-12-30 11:17 AM**. We want to change the launch time to **1:00 AM**. The campaign name and schedule ID are sent as part of the endpoint, and the request body contains the desired changes:

```
{
   "scheduleType": "ONCE",
   "scheduledTime": "2016-12-30 1:00 AM"
}
```

Sample Response Body:

The response to the above request returns the launch schedule information, showing the new `scheduledTime`.

```
{
  "id": 244797,
  "scheduleType": "ONCE",
  "scheduledTime": "2016-12-30 01:00 AM",
  "launchOptions": {
    "proofLaunch": false,
    "progressEmailAddresses": [
      "email1@ora.com",
      "email2@ora.com"
    ],
    "progressChunk": "CHUNK_10K"
  },
  "links": [
    {
      "rel": "self",
      "href": "/rest/api/v1.3/campaigns/JMP Test for
API/schedule/244797",
      "method": "PUT"
    },
    {
      "rel": "deleteSchedule",
      "href": "/rest/api/v1.3/campaigns/JMP Test for
API/schedule/244797",
      "method": "DELETE"
    },
    {
      "rel": "getSchedule",
      "href": "/rest/api/v1.3/campaigns/JMP Test for
API/schedule/244797",
      "method": "GET"
    },
    {
      "rel": "createSchedule",
      "href": "/rest/api/v1.3/campaigns/JMP Test for API/schedule",
      "method": "POST"
    }
  ]
}
```

Delete (Unschedule) an Email or Push campaign launch schedule

Use this interface to delete the launch schedule of an existing Email or Push campaign, by using the schedule ID returned from the campaign schedule.

Service URL:

`/rest/api/v1.3/campaigns/{campaignName}/schedule/{scheduleId}`

Required Path Parameters:

- `campaignName`
- `scheduleId` (this can be obtained from the id parameter from the response to either the “Get All Launch Schedules for an Email or Push Campaign” task or the “Schedule an Email or Push Campaign Launch” task)

Request Method:

`DELETE`

Request Header:

`Authorization=<AUTH_TOKEN>`

Request Body:

`None`

Sample Response Body

```
{
"id": 1491,
  "scheduleType": "ONCE",
  "scheduledTime": "2015-11-30 01:00 AM",
  "launchOptions": {
    "proofLaunch": false
  },
  "links": [
    {
      "rel": "self",
      "href": "/rest/api/v1.3/campaigns/test/schedule/1491",
      "method": "DELETE"
    },
    {
      "rel": "updateSchedule",
      "href": "/rest/api/v1.3/campaigns/test/schedule/1491",
      "method": "PUT"
    },
    {
      "rel": "getSchedule",
      "href": "/rest/api/v1.3/campaigns/test/schedule/1491",
      "method": "GET"
    },
    {
      "rel": "createSchedule",
      "href": "/rest/api/v1.3/campaigns/test/schedule",
      "method": "POST"
    }
  ]
}
```

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

Use this interface to create a content library folder in the location specified by the `folderPath` parameter in the request body.

Service URL:

`/rest/api/v1.3/clFolders`

Request Method:

`POST`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Request Body:

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

Sample Response in case of success:

```
{
   "folderPath": "/contentlibrary/abn/f1/f2/f3",
   "links":    [
      {
         "rel": "self",
         "href": "/rest/api/v1.3/clFolders",
         "method": "POST"
      },
      {
         "rel": "listContentLibraryFolders",
         "href":
"/rest/api/v1.3/clFolders/contentlibrary/abn/f1/f2/f3",
         "method": "GET"
      },
      {
         "rel": "deleteContentLibraryFolder",
         "href":
"/rest/api/v1.3/clFolders/contentlibrary/abn/f1/f2/f3",
         "method": "DELETE"
      }
   ]
}
```

Sample Response in case of failure:

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

Delete content library folder

Use this interface to delete a content library folder.

Service URL:

`/rest/api/v1.3/clFolders/{folderPath}`

Required Path Parameter:

`folderPath`

Request Method:

`DELETE`

Request Header:

`Authorization=<AUTH_TOKEN>`

Request Body:

None

Sample Response in case of success:

```
{
   "folderPath": "/contentlibrary/abn/f1/f2/f3",
   "links":    [
      {
         "rel": "self",
         "href":
"/rest/api/v1.3/clFolders/contentlibrary/abn/f1/f2/f3",
         "method": "DELETE"
      },
      {
         "rel": "createContentLibraryFolder",
         "href": "/rest/api/v1.3/clFolders",
         "method": "POST"
      }
   ]
}
```

Sample Response in case of failure:

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

List contents of a content library folder

Use this interface to list the contents of a content library folder.

Service URL:

`/rest/api/v1.3/clFolders/{folderPath}?type=<objecttype>`

Required Path Parameter:

- `folderPath` - specify the path to the folder.

  To get the objects at the root level (that is, for the entire account), replace the `folderPath` parameter with the value `contentlibrary`.

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.

Request Body:

None

Sample Response in case of success:

For the requested folder at `/contentlibrary/jmptest`, where the type was either ‘all’ or not specified, the following response was returned. The response lists the paths for all of the folders, documents, and images contained in the requested folder, and it returns links for the APIs applicable for each object.

```
{
  "folders": [
    {
      "folderPath": "/contentlibrary/jmptest/images",
      "links": [
        {
          "rel": "createContentLibraryFolder",
          "href": "/rest/api/v1.1/clFolders",
          "method": "POST"
        },
        {
          "rel": "deleteContentLibraryFolder",
          "href":
"/rest/api/v1.1/clFolders/contentlibrary/jmptest/images",
          "method": "DELETE"
        },
        {
          "rel": "listContentLibraryFolders",
          "href":
"/rest/api/v1.1/clFolders/contentlibrary/jmptest/images",
          "method": "GET"
        }
      ]
    },
    {
      "folderPath": "/contentlibrary/jmptest/testdocs",
      "links": [
        {
          "rel": "createContentLibraryFolder",
          "href": "/rest/api/v1.1/clFolders",
          "method": "POST"
        },
        {
          "rel": "deleteContentLibraryFolder",
          "href":
"/rest/api/v1.1/clFolders/contentlibrary/jmptest/testdocs",
          "method": "DELETE"
        },
        {
          "rel": "listContentLibraryFolders",
          "href":
"/rest/api/v1.1/clFolders/contentlibrary/jmptest/testdocs",
          "method": "GET"
        }
      ]
    }
  ],
  "documents": [
    {
      "documentPath": "/contentlibrary/jmptest/newsletter.htm",
      "content": null,
      "links": [
        {
          "rel": "createDocument",
          "href": "/rest/api/v1.1/clDocs",
          "method": "POST"
        },
        {
          "rel": "getDocumentContent",
          "href":
"/rest/api/v1.1/clDocs/contentlibrary/jmptest/newsletter.htm",
          "method": "GET"
        },
        {
          "rel": "deleteDocument",
          "href":
"/rest/api/v1.1/clDocs/contentlibrary/jmptest/newsletter.htm",
          "method": "DELETE"
        },
        {
          "rel": "setDocumentContent",
          "href":
"/rest/api/v1.1/clDocs/contentlibrary/jmptest/newsletter.htm",
          "method": "POST"
        }
      ]
    },
    {
      "documentPath": "/contentlibrary/jmptest/testwithimage.htm",
      "content": null,
      "links": [
        {
          "rel": "createDocument",
          "href": "/rest/api/v1.1/clDocs",
          "method": "POST"
        },
        {
          "rel": "getDocumentContent",
          "href":
```

```
"/rest/api/v1.1/clDocs/contentlibrary/jmptest/testwithimage.htm",
          "method": "GET"
        },
        {
          "rel": "deleteDocument",
          "href":
"/rest/api/v1.1/clDocs/contentlibrary/jmptest/testwithimage.htm",
          "method": "DELETE"
        },
        {
          "rel": "setDocumentContent",
          "href":
"/rest/api/v1.1/clDocs/contentlibrary/jmptest/testwithimage.htm",
          "method": "POST"
        }
      ]
    }
  ],
  "items": [
    {
      "itemPath": "/contentlibrary/jmptest/dinosmall.jpg",
      "itemData": null,
      "links": [
        {
          "rel": "getContentLibraryItem",
          "href":
"/rest/api/v1.1/clItems/contentlibrary/jmptest/dinosmall.jpg",
          "method": "GET"
        },
        {
          "rel": "setContentLibraryItem",
          "href":
"/rest/api/v1.1/clItems/contentlibrary/jmptest/dinosmall.jpg",
          "method": "POST"
        },
        {
          "rel": "createContentLibraryItem",
          "href": "/rest/api/v1.1/clItems",
          "method": "POST"
        },
        {
          "rel": "deleteContentLibraryItem",
          "href":
"/rest/api/v1.1/clItems/contentlibrary/jmptest/dinosmall.jpg",
          "method": "DELETE"
        }
      ]
    }
  ],
  "links": [
    {
      "rel": "self",
      "href": "/rest/api/v1.1/clFolders/contentlibrary/jmptest",
      "method": "GET"
    },
    {
      "rel": "deleteContentLibraryFolder",
      "href": "/rest/api/v1.1/clFolders/contentlibrary/jmptest",
      "method": "DELETE"
    }
  ]
}
```

Sample Response in case of failure:

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

Managing Content Library Documents

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

Create content library document

Use this interface to create a content library document.

Service URL:

`/rest/api/v1.3/clDocs`

Request Method:

`POST`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Sample Request Body:

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

Sample Response in case of success:

RESPONSE NOTE: Content Library saves files with an `.html` extension with an `.htm` extension. Therefore, the response will have links to the document with an `.htm` extension.

```
{
  "documentPath": "/contentlibrary/abn/wsrest_cl.htm",
  "content": null,
  "links": [
    {
      "rel": "self",
      "href": "/rest/api/v1.3/clDocs",
      "method": "POST"
    },
    {
      "rel": "getDocumentContent",
      "href": "/rest/api/v1.3/clDocs/contentlibrary/abn/wsrest_
cl.htm",
      "method": "GET"
    },
    {
      "rel": "deleteDocument",
      "href": "/rest/api/v1.3/clDocs/contentlibrary/abn/wsrest_
cl.htm",
      "method": "DELETE"
    },
    {
      "rel": "setDocumentContent",
      "href": "/rest/api/v1.3/clDocs/contentlibrary/abn/wsrest_
cl.htm",
      "method": "POST"
    }
  ]
}
```

Sample Response in case of failure:

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

Retrieve contents of a content library document

Use this interface to retrieve the contents of a content library document.

Service URL:

`/rest/api/v1.3/clDocs/{documentPath}`

Required Path Parameter:

`documentPath`

Request Method:

`GET`

Request Header:

`Authorization=<AUTH_TOKEN>`

Request Body:

None

Sample Response in case of success:

RESPONSE NOTE: The response contains the content of the document as it is saved in the content library. The Web Services API does not decode the content of the document before returning the response.

```
{
   "documentPath": "/contentlibrary/abn/wsrest_cl.htm",
   "content": "<html dir=\"ltr\">\r\n <head>\r\n
<title><\/title>\r\n <\/head>\r\n <body>\r\n  <p>test
document<\/p>\r\n  <p><img src=\"wsrest_cl.images/testcreate-
1.png\" alt=\"\" /><\/p>\r\n <\/body>\r\n<\/html>\n\n\n",
   "links":    [
       {
         "rel": "self",
         "href": "/rest/api/v1.3/clDocs/contentlibrary/abn/wsrest_
cl.htm",
         "method": "GET"
      },
      {
         "rel": "deleteDocument",
         "href": "/rest/api/v1.3/clDocs/contentlibrary/abn/wsrest_
cl.htm",
         "method": "DELETE"
      },
      {
         "rel": "setDocumentContent",
         "href": "/rest/api/v1.3/clDocs/contentlibrary/abn/wsrest_
cl.htm",
         "method": "POST"
      },
      {
         "rel": "createDocument",
         "href": "/rest/api/v1.3/clDocs",
         "method": "POST"
      }
   ]
}
```

Sample Response in case of failure:

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

Update contents of a content library document

Use this interface to update the contents of a content library document.

Service URL:

`/rest/api/v1.3/clDocs/{documentPath}`

Required Path Parameter:

`documentPath`

Request Method:

`POST`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Sample Request Body:

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

Sample Response in case of success:

RESPONSE NOTE: The content attribute in the response is returned as `null` always.

This is done to avoid returning large contents in the response.

```
{
  "documentPath": "/contentlibrary/abn/wsrest_cl.htm",
  "content": null,
  "links": [
    {
      "rel": "self",
      "href": "/rest/api/v1.3/clDocs/contentlibrary/abn/wsrest_
cl.htm",
      "method": "POST"
    },
    {
      "rel": "getDocumentContent",
      "href": "/rest/api/v1.3/clDocs/contentlibrary/abn/wsrest_
cl.htm",
      "method": "GET"
    },
    {
      "rel": "deleteDocument",
      "href": "/rest/api/v1.3/clDocs/contentlibrary/abn/wsrest_
cl.htm",
      "method": "DELETE"
    },
    {
      "rel": "createDocument",
      "href": "/rest/api/v1.3/clDocs",
      "method": "POST"
    }
  ]
}
```

Sample Response in case of failure:

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

Delete a content library document

Use this interface to delete a content library document.

Service URL:

`/rest/api/v1.3/clDocs/{documentPath}`

Required Path Parameter:

`documentPath`

Request Method:

`DELETE`

Request Header:

`Authorization=<AUTH_TOKEN>`

Request Body:

None

Sample Response in case of success:

```
{
   "documentPath": "/contentlibrary/abn/wsrest_cl.htm",
   "content": null,
   "links":    [
     {
         "rel": "self",
         "href": "/rest/api/v1.3/clDocs/contentlibrary/abn/wsrest_
cl.htm",
         "method": "DELETE"
      },
      {
         "rel": "createDocument",
         "href": "/rest/api/v1.3/clDocs",
         "method": "POST"
      }
   ]
}
```

Sample Response in case of failure:

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

Create a copy of a content library document

Use this interface to create a copy of a content library document.

Service URL:

`/rest/api/v1.3/clDocs/{destinationDocumentPath}`

Request Method:

`PUT`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Request JSON Body:

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

Sample Response in case of success:

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

```
{
   "documentPath": "/contentlibrary/abn/doc22.htm",
   "content": null,
   "links":    [
     {
         "rel": "self",
         "href":
"/rest/api/v1.3/clDocs/contentlibrary/abn/doc22.htm",
         "method": "PUT"
      },
      {
         "rel": "getDocumentContent",
         "href":
"/rest/api/v1.3/clDocs/contentlibrary/abn/doc22.htm",
         "method": "GET"
      },
      {
         "rel": "deleteDocument",
         "href":
"/rest/api/v1.3/clDocs/contentlibrary/abn/doc22.htm",
         "method": "DELETE"
      }
   ]
}
```

Sample Response in case of failure:

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

Managing Content Library Media Files

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

Create content library media file

Use this interface to create a content library media file in a content library folder.

Service URL:

`/rest/api/v1.3/clItems`

Request Method:

`POST`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Sample Request Body:

REQUEST NOTE: Ensure that the `itemData` value is a base64-encoded binary string with no extraneous hidden characters. When the `itemData` value contains hidden characters, such as such as carriage returns or line feeds, the response returns an HTTP code of 500 and the error details return only an ID number.

```
{
  "itemPath": "/contentlibrary/abn/testcreate_50.png",
  "itemData": "<base64 encoded binary string>"
}
```

Sample Response in case of success:

RESPONSE 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/v1.3/clItems",
         "method": "POST"
      },
      {
         "rel": "getContentLibraryItem",
         "href":
"/rest/api/v1.3/clItems/contentlibrary/abn/testcreate_50.png",
         "method": "GET"
      },
      {
         "rel": "deleteContentLibraryItem",
         "href":
"/rest/api/v1.3/clItems/contentlibrary/abn/testcreate_50.png",
         "method": "DELETE"
      },
      {
         "rel": "setContentLibraryItem",
         "href":
"/rest/api/v1.3/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

Use this interface to retrieve the contents of a content library media file.

Service URL:

`/rest/api/v1.3/clItems/{itemPath}`

Required Path Parameter:

`itemPath`

Request Method:

`GET`

Request Header:

`Authorization=<AUTH_TOKEN>`

Request Body:

`None`

Sample Response in case of success:

RESPONSE NOTE: The `itemData` attribute in the response is a BASE64 encoded binary string representation of the media file content.

```
{
   "itemPath": "/contentlibrary/abn/testcreate_50.png",
   "itemData":
"iVBORw0KGgoAAAANSUhEUgAAAdwAAAFUCAIAAAABIhoXAAAAAXNSR0IArs4c6QAAAA
RnQU1BAACxjwv8YQUAAAAJcEhZcwAAFiUAABYlAUlSJPAAAA6XSURBVHhe7d0vVBtLG
8BhZCQyEhmJRCKRkUgkEhkXiYy8MhKJREYiI5GRkZHIfnO60zn7hT9N20z2TfZ5xD3N
LoWEc/tjmMzOnv0AIAxRBghElAECEWWAQEQZIBBRBghElAECEWWAQEQZIBBRBghElAE
CEWWAQEQZIBBRBghElAECEWWAQEQZIBBRBghElAECEWWAQEQZIBBRBghElAECEWWAQE
QZIBBRBghElAECEWWAQEQZIBBRBghElAECEWWAQEQZIBBRBghElAECEWWAQEQZIBBRB
ghElAECEWWAQEQZIBBRBghElAECEWWAQEQZIBBRBghElAECEWWAQEQZIBBRBghElAEC
EWWAQEQZIBBRBghElAECEWWAQEQZIBBRBghElAECEWWAQEQZIBBRBghElAECEWWAQEQ
ZIBBRBghElAECEWWAQEQZIBBRBghElAECEWWAQEQZIBBR7sDLy8v19fV8Ps+PAX4R5Q
4Mh8Ozs7PBYJAfA/wiyh1IOU5RTD+dHjrNUTWIq/9D9/f3c/Kw4yXs+wcycSRnc6U/7+f3
exssVjkE/uz2WweHx8vLi7y1ziU9BXv7+rvCLoCVHuRury9fV1CVl6mE/8s9VqlbJYD
Zkg6l17df//9l58c8Dui3Jn1en1+ft6UK0H/0HaXw6Ho+bTxiNNNMOORLlL8/k8R+v
sbLlc5qN/Ln2ey8vL/Il+SSPx5+fn/BGHkl7Fw8PDV9MmJtDht0S5Y2VyVqonx/Gd6O
VsFti13d3d/Uvi9+j9/T09mfy0fjFqhm+IcsdyWofAe7zz1ZVFmOe1ARPkaHkV
ZgkN1l7df//9l58c8Dui3Jn1en1+ft6UK2U0H/0HaXw6Ho+bTxiNNMOORLlL8/k8R+v
sbLlc5qN/Ln2ey8vL/Il+SSPx5+fn/BGHkl7Fw8PDV9MmJtDht0S5Y2VyOQ1y86GdNQ
VsFti13d3d/Uvi9+j9/T09mfy0fjFqhm+IcsdSPXOrdh4sfzUaPT8/n0wm6/U6f1wYW
29sFuklpBcS8AlDh0S5e2Ui+PvB8lctTpqx5x7fLdy7T4fMbYbP0BDl7v12sPzbh
cHh068++r7NJZxDlEL4aLL+9vV1dXTWnmHzbjBNs8aeen58/vj+GRUeamXOdGRt2u
KeWovNz6ZFn/UnnS2JQg9J8pR3N7eliqGlx4BBRLHZbD
6dOE51fnt7yx900vILFmX6zT+AQF5fX9vzFX0Y6RPT6TS/bvBh8VG3N0vVPH
c1HoZdEOYqtIidppNyHiJYvJRrrGX5YjHLWnGQxmp8cLTExJ1ZGj3SW2Kn8xZrDkKE
```

```
MiSi3L2tIqcwtSeX7+7u8sedlvl83p5AV2RoiHLHPha5OT6bzfLRs7PTu/64vT1er97
PhN8S5S6l2uYyfTZULJf5pWylsXM+evzaRe7Pgj/YkSh3qcwdf/rLezpSrkIeDoensZ
tau8imLOAjUe5MimzTpvPz86/atFqtyt1J0qDy2BOmyPBbotyZ5+fnJk/X19f50GcWi
0X7Tb/jDZkiwy5EuTPlLb6Hh4d86AvtN/1Go9Exzi8rMuxIlDtT3sd7enrKh752f3/f
fHBjMpnkE8eg/UNFkeF7otyZMlm84/KDNNgsfyW5vr7ebDb5XGDp94D8jBUZdiDK3Sg
bKKfO5kM7WK/XqWvNX0wuLi4+vVNJEKm/ZT/SJP1moMjwW6LcjfIb/V9csNe+3iQ1/f
n5OZ+IJI3i01g+P8uzs/v7+3wC+JYod6MMeOe/7jPyR1KI21MZk8kk1CB0tVq17/Pkg
j3YnSh3IAW0rHJL/cpH/9Db29toNGo+SXJxcRFkyLxcLstFMell/t1PHegtUe7AYrFo
mpXilQ/9lc1mU5ZwNNIAvNurltNLK0P49IeXl5d8AtiNKHegTArvZab16elpOBw2n7D
RyWxG+oqPrTu9pqcU+U1ICEuUO1DmW/c14ZCGzO2t4pMDX2Myn8/bPxjSV//raRnoOV
E+tBTQnK6zs/0uNH57e2sveEgOMGReLBbt9/SSq6ur/b4u6BVRPrSy5UVqWT60V1vXm
NQbMqefAe1F00kaLJ/e1s9wYKJ8aHd3d03C6l0qvXWNSXJ7e5tyua/3ANPn37rsezAY
TKfTUMvy4EiJ8qGV5WLpF/98qI6tIXM9KdCnsdczRCDKB7VarZqQpaHlAcaVH4fM+5U
+uSUWsF+ifFDl/k/f76G8X6mbs9lsPB7vceB8eXlpDTLUIMoHVa71mE6n+RBAiygfzv
v7exmr+q0f+JQoH05ZDDccDvMhgP8nyodTNheutxgOOHaifCCbzabsC2HuAviKKB/I/
NedQ0ejUT4E8IEoH0jZlcKO78A3RPkQ1ut1U+TE9mnAN0T5ENLouCnyIa8ZAY6RKB9C
2dzSJmrA90S5ure3t6bIg8HARsPA90S5unJPkPF4nA8BfEGUqyt7dQa52zQQmSjXVW5
cfX5+bg944LdEua5yh4693LgaOHmiXNfB7jMCnAZRrqvceD8/BviWWNTVFDnJjwG+JR
Z1NUW+uLjIjwG+JcoVldukivLfmc/n19fX6b/5MfSAKFdUonx5eZkPsbPpdNp89w5z5
28IQpQrWi6XTVbsQ/Snym2/k9vb23wUekCUKypXjtzc3ORD7GC9Xpe7tKRvnWEyvSLK
Fb28vDRlubu7y4fYwcPDQ/N9u7y8VGT6RpQrKreAEuXdtYfJdguhh0S5ohJlt6/eXfo
B1nzTvDtKP4lyReWGI9PpNB/iay8vL6PRqPmOJYbJ9JMoV1QWdc1ms3yIzyyXy3Jj2Y
a3RuktUa6ovGHl8oevrNfrspFeYzgcumkWfSbKFZXpUVH+1OPj4/n5efMtSgaDwWQyc
ccsek6UKypRtm/nlre3t6urq+ab0xiPx6vVKp+GHhPliso8qSi3pQFyWfSWXF5e+v5A
IcoVlSingWE+1G/r9bo9QE5pToHO54CfRLmissDLL+bJfD5vzyCnOvtZBR+JckXlXlB
phJgP9VJ6+Tc3N823IjFAhm+IckWlQflxL81mMwNk2J0o11I2Ux6NRvlQz2xdoWeADL
sQ5Vr6vG9nGgu35ysSA2TYkSjXUnYjur+/z4d6YLPZlOsSYG67Qgz8iyrWUjS/68zv71
vTxwBV68OdEuZZyOd/T01M+dLq2po8TV+jB3xHlWsqVI6+vr/nQKfo4fewKPfgXolzL
yS9SNn0MNYhyLU2nTnWR8tPTk+ljqEGUqzjtRcqpv82ra5g+hj0S5SpOdZHy+/v77e1
t89KS9CPH9DHslyhXcZKLlDebTfPSksvLSz/HJnXVatVe9Pbw8JBHslyl
Wc2CLl19fX9tt6llhAPaJcxSktUk4/Vwa/bhSS0vzy8pJPABWIchUns0h5Nps1LyRJL
8qmQlCbKFdRopwfH6f2tSFXV1fe1oMDEOUqjj3KW0vfxuNxOpLPATWJchVHHeU0Ii5z
4kmvth6FzolyFccb5dVqdXl52Tz5xL1C4MBEuYojjfJyuSzPfDAYzFfzAI4FFGu4hi
jvFgsymJkS9+gK6JcxdFFeTablcwCwRLmKI4ry1k1D0p9t+QYdEuUqji
LKn95z2mJk6JYoVxE8yqm8bhoCMYlyFZGjvHXP6cRNQjAIUa4i
ihXUYai+XHXlsvl1vRxcnNz46Qh6CZR7uHmUnZKs9c5XsePdhOK4ngTMjwQ
XUf54/Txw8OD6WOISryAnsLsqpvFvzFaaP4SiIchU5hB1FebFYDIfD/AxMH8NREeUe
qcg67iPJkMslf+6f7+3tbIcMREeUqchEPG+XVanV1dW/d9NOs3jVW/8M839J6fn/M54EiIchW5iw
eM8tPTU/s9vVTmY0PSTKFeR03i3iQKL+KKdc/9N0Os3ngzGMjylXkXkkxOhlsPGM9Ozq5yFF3WPRaYBm4uLD
F+kNxwOX19f8zngCIlyFbmRlaM8n8/LJsLJ6eWxMk5ksk6cvZ
4w1OgyhXUTbGTLmsMZ+wXC5L90ejkatC4GSIchXv7+9lddpwONzvQojValWuDbm4uLD
KAk6JKNfSTmcK9L6u4NhsNuWdvfPzc2NkODGiXNFisSiTDHd3d//e5fYAPH3m9PnzCe
BUiHJds9msaWiSRrj/OL88Ho/z5zo7c7UenCRRrm5rO+PHx8d84g+176pnQ2Q4VaJ8C
PP5vH0NdHqYT+xsOp3mv/xzT+R8FDg5onwg6/X6+vq6qepgMPijN+jaRR6Px/kocIpE
+XA2m025oepoNNrxfb92kW9ubva1igOISZQP6vX1tb0eIx/9miJD34jyobXXY3x/QxB
Fhh4S5Q6UlW3D4fCrLYQUGfpJlDuwXq/b10l0PJpOt5ioy9JYod+Pp6SlH96fmupLVaj
```

```
Wfz29vb/NRRYb+EeXObN3h9CNFhh4S5Y5tXVdSKDL0kyh3b71epwSnEA8Gg/SHx8dHt
3SC3hJlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFl
gEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSA
QUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJ
QBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlg
EBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQ
UQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQ
BAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgE
BEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQU
QYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQB
AhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgEBEGSAQUQYIRJQBAhFlgDB
+/PgfhFa/3C47xMoAAAAASUVORK5CYII=",
   "links":    [
            {
         "rel": "self",
         "href":
"/rest/api/v1.3/clItems/contentlibrary/abn/testcreate_50.png",
         "method": "GET"
      },
            {
         "rel": "deleteContentLibraryItem",
         "href":
"/rest/api/v1.3/clItems/contentlibrary/abn/testcreate_50.png",
         "method": "DELETE"
      },
            {
         "rel": "setContentLibraryItem",
         "href":
"/rest/api/v1.3/clItems/contentlibrary/abn/testcreate_50.png",
         "method": "POST"
      },
            {
         "rel": "createContentLibraryItem",
         "href": "/rest/api/v1.3/clItems",
         "method": "POST"
      }
   ]
}
```

Sample Response in case of failure:

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

Update contents of a content library media file

Use this interface to update the contents of a content library media file.

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

Service URL:

`/rest/api/v1.3/clItems/{itemPath}`

Required Path Parameter:

`itemPath`

Request Method:

`POST`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Sample Request JSON Body:

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

REQUEST NOTE: Ensure that the `itemData` value is a base64-encoded binary string with no extraneous hidden characters. When the `itemData` value contains hidden characters, such as such as carriage returns or line feeds, the response returns an HTTP code of 500 and the error details return only an ID number.

Sample Response in case of success:

RESPONSE 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/v1.3/clItems/contentlibrary/abn/testcreate_50.png",
         "method": "POST"
      },
            {
         "rel": "getContentLibraryItem",
         "href":
"/rest/api/v1.3/clItems/contentlibrary/abn/testcreate_50.png",
         "method": "GET"
      },
            {
         "rel": "deleteContentLibraryItem",
         "href":
"/rest/api/v1.3/clItems/contentlibrary/abn/testcreate_50.png",
         "method": "DELETE"
      },
            {
         "rel": "createContentLibraryItem",
         "href": "/rest/api/v1.3/clItems",
         "method": "POST"
      }
   ]
}
```

Sample Response in case of failure:

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

Delete a content library media file

Use this interface to delete a content library media item.

Service URL:

`/rest/api/v1.3/clItems/{itemPath}`

Required Path Parameter:

`itemPath`

Request Method:

`DELETE`

Request Header:

`Authorization=<AUTH_TOKEN>`

Request Body:

`None`

Sample Response in case of success:

```
{
   "itemPath": "/contentlibrary/abn/testcreate_50.png",
   "itemData": null,
   "links":    [

            {
         "rel": "self",
         "href":
"/rest/api/v1.3/clItems/contentlibrary/abn/testcreate_50.png",
         "method": "DELETE"
      },
            {
         "rel": "createContentLibraryItem",
         "href": "/rest/api/v1.3/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

Use this interface to create a copy of a content library media file.

Service URL:

`/rest/api/v1.3/clItems/{destinationItemPath}`

Required Path Parameter:

`destinationItemPath`

Request Method:

`PUT`

Request Header:

```
Authorization=<AUTH_TOKEN>

Content-Type=application/json
```

Sample Request JSON Body:

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

Sample Response in case of success:

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

```
{
  "itemPath": "/contentlibrary/abn/copiedimage.png",
  "itemData": null,
  "links": [
    {
      "rel": "self",
      "href":
"/rest/api/v1.3/clItems/contentlibrary/abn/copiedimage.png",
      "method": "PUT"
    },
    {
      "rel": "getContentLibraryItem",
      "href":
"/rest/api/v1.3/clItems/contentlibrary/abn/copiedimage.png",
      "method": "GET"
    },
    {
      "rel": "deleteContentLibraryItem",
      "href":
"/rest/api/v1.3/clItems/contentlibrary/abn/copiedimage.png",
      "method": "DELETE"
    }
  ]
}
```

Sample Response in case of failure:

The system sends the following response when it cannot find the source folder **or** file or the destination folder:

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

Sample Response in case of failure - File type mismatch:

For a request to copy a file `dinoSmall.jpg` (source file type JPEG) to `/contentlibrary/jmptest/testdocs/copyDinoSmall2.png` (destination file type PNG), the system returns the following response:

```
{
  "type": "",
  "title": "Invalid request parameters",
  "errorCode": "INVALID_PARAMETER",
  "detail": "Invalid destination
:/contentlibrary/jmptest/testdocs/copyDinoSmall2.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

Use this interface to set images in a content library document.

Service URL:

`/rest/api/v1.3/clDocImages/{documentPath}`

Required Path Parameter:

`documentPath`

Request Method:

`POST`

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Request Body Parameters

`documentPath`	string	The complete path of the document, starting with /contentlibrary.
`imageData`	array	Images in the content library document. Contains itemData and itemPath for each image in the document.
`itemPath`	string	Name of the image in the content library document.
`itemData`	string (byte)	Base64 encoded binary string of the image content.

Sample Request JSON Body:

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

Response Notes:

- A success response echoes back the document path you sent, and it returns content as null. It also returns the related endpoints for the given document path.
- Possible error responses occur when:
  - The `documentPath` is incorrect in the request (error codes include `FOLDER_NOT_FOUND` and `DOCUMENT_NOT_FOUND` and the details show the folder path without the file name). Verify that the folder and file names are spelled correctly and that they exist in the Content Library. To get the correct path, you can ask the Responsys Account Admin to send you a copy of the document path, which the admin can get from the Content Library.
  - The `documentPath` has incorrect format in the request (error code is `INVALID_PARAMETER` and the details show the folder path without the file name). Verify that you have included `/contentlibrary` at the start of the path, have spelled it correctly, and have included the document file name at the end of the path.

Sample Response in case of success:

```
{
   "documentPath": "/contentlibrary/abn/wsrest_cl.htm",
   "content": null,
   "links":    [
            {
         "rel": "self",
         "href":
"/rest/api/v1.3/clDocImages/contentlibrary/abn/wsrest_cl.htm",
         "method": "POST"
      },
            {
         "rel": "getDocumentImages",
         "href":
"/rest/api/v1.3/clDocImages/contentlibrary/abn/wsrest_cl.htm",
         "method": "GET"
      }
   ]
}
```

Sample Response in case of failure:

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

Get images in a content library document

Use this interface to get images in a content library document.

Service URL:

`/rest/api/v1.3/clDocImages/{documentPath}`

Required Path Parameter:

`documentPath`

Request Method:

`GET`

Request Header:

`Authorization=<AUTH_TOKEN>`

Request Body:

None

Response Notes:

- A success response echoes back the document path requested, and it returns the path and Base64-encoded binary string content for each image in the document.
- Possible error responses occur when:
  - The `documentPath` is incorrect in the request (error codes include `FOLDER_NOT_FOUND` and `DOCUMENT_NOT_FOUND` and the details show the folder path without the file name). Verify that the folder and file names are spelled correctly and that they exist in the Content Library. To get the correct path, you can ask the Responsys Account Admin to send you a copy of the document path, which the admin can get from the Content Library.
  - The `documentPath` has incorrect format in the request (error code is `INVALID_PARAMETER` and the details show the folder path without the file name). Verify that you have included `/contentlibrary` at the start of the path, have spelled it correctly, and have included the document file name at the end of the path.
  - The document in your request does not contain images, or the path to the image is broken (error code is `IMAGES_NOT_FOUND`). Verify that you are requesting the images for the correct Content Library file. You can also retrieve the HTML document from the content library and examine it to determine if there are errors in the image tags, such as the wrong file extension, file name, or path to the image file.

Sample Response in case of success:

```
{
   "documentPath": "/contentlibrary/abn/wsrest_cl.htm",
   "imageData": [   {
      "itemPath": "/contentlibrary/abn/testcreate.png",
      "itemData": "<base64 encoded binary string>",
      "links":       [
                  {
            "rel": "getContentLibraryItem",
            "href":
"/rest/api/v1.3/clItems/contentlibrary/abn/testcreate.png",
            "method": "GET"
         },
                  {
            "rel": "setContentLibraryItem",
            "href":
"/rest/api/v1.3/clItems/contentlibrary/abn/testcreate.png",
            "method": "POST"
         },
                  {
            "rel": "deleteContentLibraryItem",
            "href":
"/rest/api/v1.3/clItems/contentlibrary/abn/testcreate.png",
            "method": "DELETE"
         },
                  {
            "rel": "createContentLibraryItem",
            "href": "/rest/api/v1.3/clItems",
            "method": "POST"
         }
      ]
   }],
   "links":    [
            {
         "rel": "self",
         "href":
"/rest/api/v1.3/clDocImages/contentlibrary/abn/wsrest_cl.htm",
         "method": "GET"
      },
            {
         "rel": "setDocumentImages",
         "href":
"/rest/api/v1.3/clDocImages/contentlibrary/abn/wsrest_cl.htm",
         "method": "POST"
      }
   ]
}
```

Sample Response in case of failure:

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

REST API v1.4 resources

Create a new profile extension table

You can create a new profile extension table for a given profile list by providing the schema of the profile extension table.

Service URL:

`/rest/api/v1.4/lists/{listName}/listExtensions`

Request Method:

`POST`

Request Body Properties:

- `objectName` - name of the Profile List.
- `folderName` - the name of the folder the Profile List is located.
- `expiryDays` - number of days the profile extension table will exist before expiring.
- `fields` - the profile extension table fields. When creating a new profile extension table, the supported field types are:
  - `STR500`
  - `STR4000`
  - `INTEGER`
  - `NUMBER`
  - `TIMESTAMP`

Sample Request Body:

```
{
  "profileExtension": {
    "objectName": "ws_rest_petxx",
    "folderName": "WS_REST_SAMPLE",
    "expiryDays": "10"
  },
  "fields": [
    {
      "fieldName": "edu",
      "fieldType": "STR500"
    }
  ]
}
```

Request Header:

`Authorization=<AUTH_TOKEN>`

`Content-Type=application/json`

Response if successful:

```
true
```

Sample Response if failure:

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

Responsys API Data Types

The Responsys API uses the standard data types defined below. These data types conform to their specifications in the World Wide Web Consortium's publication “XML Schema Part 2: Data Types” (available at http://www.w3.org/TR/xmlschema-2). Data types are used as a standardized way to define, send, receive, and interpret basic data types in the request/response messages exchanged between client applications and the Responsys API.

Type	Description
boolean	Boolean fields have one of these values: true (or 1), or false (or 0).
string	Character string data types contain text data. In some cases, strings are enumerated; that is, the text data values are restricted to a specific set of expected values.
int and long	Fields of these types contain integers (long ranges from 9223372036854775807 to -9223372036854775808 and int ranges from 2147483647 to -2147483648.
dateTime	Fields defined as dateTime data types handle date/time values (timestamps). Regular dateTime fields are full timestamps with a precision of one second.

Definitions of Rule Parameters for Merging Members into a Profile List

Name	Type	Description
`insertOnNoMatch`	boolean	Indicates what should be done for records where a match is not found (`true` = insert / `false` = no insert).
`updateOnMatch`	string (enum)	Controls how the existing record should be updated. Valid values: • `NO_UPDATE` • `REPLACE_ALL`
`matchColumnName1`	string (enum)	First match column for determining whether an insert or update should occur. Valid values: • `RIID_` • `CUSTOMER_ID_` • `EMAIL_ADDRESS_` • `MOBILE_NUMBER_` • `EMAIL_MD5_HASH_` • `EMAIL_SHA256_HASH_`
`matchColumnName2`	string (enum)	Second match column for

Name	Type	Description
		determining whether an insert or update should occur (optional). Valid values: • `null` • `RIID_` • `CUSTOMER_ID_` • `EMAIL_ADDRESS_` • `MOBILE_NUMBER_` • `EMAIL_MD5_HASH_` • `EMAIL_SHA256_HASH_`
`matchColumnName3`	string	**DO NOT USE.** This attribute is no longer supported, but you may still see it in the response body of some APIs. For backward compatibility, it may be present in the request body but it must be set to null.
`matchOperator`	string (enum)	Controls how the Boolean expression involving the match columns is constructed to determine a

Name	Type	Description
		match between the incoming records and existing records. Valid values: • NONE • AND
`optinValue`	string (enum)	Value of incoming opt-in status data that represents an opt-in status. For example, "`1`" may represent an opt-in status.
`optoutValue`	string	Value of incoming opt-out status data that represents an opt-out status. For example, "`0`" may represent an opt-out status.
`defaultPermissionStatus`	string (enum)	This value must be specified as either `OPTIN` or `OPTOUT` and would be applied to all of the records contained in the API call. If this value is not specified explicitly, then it is set to `OPTOUT`. Valid values: • `OPTIN` • `OPTOUT`
`htmlValue`	string	Value of incoming preferred email format data. For example, "`H`" may represent a preference for HTML

Name	Type	Description
		formatted email.
`textValue`	string	Value of incoming preferred email format data. For example, "`T`" may represent a preference for Text formatted email.
`rejectRecordIfChannelEmpty`	string	String containing comma-separated channel codes that if specified will result in record rejection when the channel address field is null. Channel codes are as follows: `E` - Email `M` - Mobile `P` - Postal address For example "E,M" would indicate that a record that has a null for Email or for Mobile Number value should be rejected. This parameter can also be set to `null` or to an empty string. By specifying `null` or an empty string, this validation will not be

Name	Type	Description
		performed for any channel, unless overridden by the matchColumnName1 setting. When matchColumnName1 is set to EMAIL_ADDRESS_ or MOBILE_NUMBER_, then the null or empty string setting is effectively ignored for that channel. For example, if a merge rule has matchColumnName1 set to EMAIL_ADDRESS_, the system will reject a record without an email address, even if the rejectRecordIfChannelEmpty is set to null.

What are the HTTPs status codes?

This topic provides information about the HTTPs status codes that can be returned from Responsys. It also provides troubleshooting information

HTTPs Status Codes

HTTP Status Code	Cause and Action
200 OK	The request was successfully completed. The system returns a 200 status for a successful GET or POST method. Action: Although you may receive a successful HTTP status code, the intended action may not have completed successfully for all records in the request body. Examine the response body to determine whether the action completed successfully for each record sent.
400 Bad Request	Cause: The system cannot fulfill the request due to incorrect syntax. Related error code examples: INVALID_PARAMETER Actions: • View the response body details to see if there is additional information about the problem. • Verify that the URL format and parameters are correct for the API call. Check whether there are required parameters and make sure they contain valid values. • Verify that the header has the correct format. • Verify that the request body is formatted correctly, if applicable.

HTTP Status Code	Cause and Action
401 Unauthorized	Cause: The system cannot fulfill the request because the user does not have sufficient access to perform the call. Related error code examples: API_DISABLED_FOR_USER, INSUFFICIENT_ACCESS, API_LIMIT_EXCEEDED Actions: • View the response body details to see if there is additional information about the problem. • Verify that the API user has the correct Roles assigned. You may need to ask the Responsys Account Administrator for assistance. • Your API code may be attempting an API call that is not part of the standard Responsys API. View the Responsys documentation to verify whether the call is standard. • For campaign-related APIs, ensure that your account is enabled for Email Message Designer (EMD) for email campaigns and Push for Push campaigns. • For Email campaign-related APIs, ensure that the campaign that you are trying to use is not a "Classic" campaign.
404 Not Found	Cause: The system cannot find the item specified in the request. Related error code examples: FOLDER_NOT_FOUND, DOCUMENT_NOT_FOUND, IMAGE_NOT_FOUND Actions: • View the response body detail to see if there is additional information about

HTTP Status Code	Cause and Action
	the problem. For example, for content library calls, it will show the document path in the request. • For API calls that include path names, verify that the path exists in the system and that the parts of the path are spelled correctly. • For image API calls/requests, examine the document file to ensure that the image tags are correct and to determine whether the document contains images.
405 Method Not Allowed	Cause: The method is incorrect for the endpoint you are using, or the endpoint format is incorrect for the method. Related error code examples: METHOD_NOT_SUPPORTED Actions: • Verify that you have used the correct method and endpoint for the task you are trying to perform. For example, if you use the GET method to obtain a content library image but do not provide the complete path in the endpoint, you will see this error.
500 Internal Server Error	Cause: The server encountered an unexpected error that prevented it from fulfilling the request. Related error code examples: UNEXPECTED_EXCEPTION, UNRECOVERABLE_EXCEPTION Actions: • View the response body to see if it provides additional information about the problem.

HTTP Status Code	Cause and Action
	• Verify that you are using the correct endpoint for the correct request body. For example, if you use the old v1.1 endpoint with the new v1.3 request format, you will receive an internal server error. • AFTM-enabled accounts only: If the error code is UNRECOVERABLE_ EXCEPTION and your code is using AFTM supported APIs, re-log in and try the call or request again
503 Service Unavailable	Cause: The server is currently unable to handle the request, due to a temporary overloading or server maintenance. Actions: • View the response body to see if it provides additional information about the problem. For example, the response may return an HTML page informing you that the Responsys service is temporarily unavailable. • Check the System Status blog page for your Responsys Interact environment to determine if there is system maintenance, and reschedule your API activity accordingly.

What are the error status codes?

The error status codes are what Responsys returns when an API request or call fails. This topic describes what to expect for REST error responses. It also provides a table of error codes with possible causes and troubleshooting actions for you to try.

Handling REST API error responses

If an API request fails, Responsys returns the following error information instead of the expected successful response. The format is as follows:

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

Where:

- *<ERROR_TITLE>* is the short description of the error.
- *<ERROR_CODE>* is the error code. Responsys error code format is descriptive text, all caps, and use underscores instead of spaces.
- *<DETAIL_MESSAGE>* provides additional details about the error.
- The parameters `type` and `errorDetails` do not return values. We have reserved them for future use.

Example:

```
{
       "type": "",
       "title": "Invalid request parameters",
       "errorCode": "INVALID_PARAMETER",
       "detail": "FieldList is null OR empty",
       "errorDetails": []
}
```

Error Status Codes

When an API call/request fails and when Responsys can determine the problem, it will return one of the following error codes. This list is organized alphabetically.

Error Code	Details
ACCOUNT_ SUSPENDED	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Account is in a suspended state. Actions: Contact the Responsys Account Administrator. The Account Administrator should contact the Responsys account's Oracle Customer Success Manager for assistance.
API_LIMIT_ EXCEEDED	HTTP response code: HttpStatus.BAD_REQUEST Causes: Request limit exceeded.
API_BLOCKED	HTTP response code: HttpStatus.BAD_REQUEST Causes: The *function_name* is currently not available to this user.
API_DISABLED_ FOR_USER	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Your account may not have permission to use the API call that you have attempted.

Error Code	Details
	Actions: If the API call is not documented in the standard API reference documentation, your code may have attempted to use a call that is typically restricted to Oracle Internal use only. You may need to refactor your code if your account is not authorized to use the call. You may also see this message if: • Your account lacks a required setting. For example, you may see this message if you are using a campaign-related call and your account is not enabled for Email Message Designer (EMD) for email campaigns or Push for mobile app campaigns. • You try to schedule an email campaign and the email campaign is a classic campaign. (EMD-enabled accounts may still force a campaign to save as classic, but this will cause issues for campaign scheduling APIs.)
API_NOT_ ALLOWED_IN_ SECONDARY	HTTP response code: HttpStatus.UNAUTHORIZED Causes: API is not allowed in secondary. Actions: Non HATM endpoint should be used.
AUTHENTICATION_ FAILED	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Authentication failed.
CAMPAIGN_ ALREADY_EXISTS	HTTP response code: HttpStatus.BAD_REQUEST Causes: Campaign already exists.
CAMPAIGN_IS_ INVALID	HTTP response code: HttpStatus.BAD_REQUEST Causes: Not a valid campaign.

Error Code	Details
CAMPAIGN_ LAUNCH_ ALREADY_ HAPPENED	HTTP response code: HttpStatus.BAD_REQUEST Causes: Campaign launch has already occurred.
CAMPAIGN_ LAUNCH_ SCHEDULE_DATE_ PAST	HTTP response code: HttpStatus.BAD_REQUEST Causes: Campaign launch date is past.
CAMPAIGN_NOT_ FOUND	HTTP response code: HttpStatus.NOT_FOUND Causes: Campaign not found.
CAMPAIGN_NOT_ LISTENING	HTTP response code: HttpStatus.BAD_REQUEST Causes: Campaign not listening.
CAMPAIGN_ SCHEDULE_ DUPLICATE	HTTP response code: HttpStatus.BAD_REQUEST Causes: Campaign is already scheduled for launch.
CAMPAIGN_ SCHEDULE_NOT_ FOUND	HTTP response code: HttpStatus.BAD_REQUEST Causes: Campaign Schedule not found.
CLIENT_ CERTIFICATE_ EXPIRED	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Client certificate expired.
CLIENT_ CERTIFICATE_ NOT_YET_VALID	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Client certificate not valid.
CLIENT_ CERTIFICATE_ NOT_FOUND	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Client certificate not found.
CURRENT_ CAMPAIGN_ SCHEDULE_AT_ SAME_TIME	HTTP response code: HttpStatus.BAD_REQUEST Causes: This Schedule is already scheduled to run at specified time.

Error Code	Details
CUSTOM_EVENT_ NOT_FOUND	HTTP response code: HttpStatus.NOT_FOUND Causes: Custom event not found.
DATA_SOURCE_ NOT_FOUND	HTTP response code: HttpStatus.NOT_FOUND Causes: Data source not found.
DOCUMENT_ ALREADY_EXISTS	HTTP response code: HttpStatus.BAD_REQUEST Causes: Document already exists.
DOCUMENT_NOT_ FOUND	HTTP response code: HttpStatus.NOT_FOUND Causes: Document not found.
DUPLICATE_API_ REQUEST	HTTP response code: HttpStatus.BAD_REQUEST Causes: Duplicate API request.
DUPLICATE_DATA_ SOURCE	HTTP response code: HttpStatus.BAD_REQUEST Causes: Duplicate data source.
FOLDER_ ALREADY_EXISTS	HTTP response code: HttpStatus.BAD_REQUEST Causes: Folder already exists.
FOLDER_NOT_ FOUND	HTTP response code: HttpStatus.NOT_FOUND Causes: Folder not found.
INACTIVE_ ACCOUNT	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Account is not active. Actions: Contact the Responsys Account Administrator. The Account Administrator should contact the Responsys account's Oracle Customer Success Manager for assistance.

Error Code	Details
IMAGES_NOT_ FOUND	HTTP response code: HttpStatus.NOT_FOUND Causes: The system cannot find any images in the requested document file, or the path to the image is broken. Actions: • Review the errorDetails text to determine the invalid parameter that caused the error message. • If you expected to receive image data in your response, verify that you are requesting the images for the correct Content Library document file. • You can also retrieve the HTML document from the content library and examine it to determine if there are errors in the image tags, such as the wrong file extension, file name, or path to the image file.
INSUFFICIENT_ ACCESS	HTTP response code: HttpStatus.UNAUTHORIZED Causes: The API user lacks one or more roles needed to perform the action. Actions: Ask the Responsys Account Administrator to verify the roles assigned to the API user.
INVALID_AUTH_ OPTION	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Not a valid authentication option.
INVALID_ AUTHENTICATION_ OPTION	HTTP response code: HttpStatus.BAD_REQUEST Causes: Not a valid authentication option.
INVALID_ CAMPAIGN_ SCHEDULE_TIME	HTTP response code: HttpStatus.BAD_REQUEST

Error Code	Details
	Causes: Invalid Schedule Time.
INVALID_ CAMPAIGN_ SCHEDULE_TYPE	HTTP response code: HttpStatus.BAD_REQUEST Causes: Invalid Campaign Schedule Type.
INVALID_ CAMPAIGN_ SCHEDULE_TYPE_ CHANGE	HTTP response code: HttpStatus.BAD_REQUEST Causes: Campaign Schedule Type cannot be changed.
INVALID_DATE	HTTP response code: HttpStatus.BAD_REQUEST Causes: Invalid date.
INVALID_FIELD_ NAME	HTTP response code: HttpStatus.BAD_REQUEST Causes: Invalid field name.
INVALID_NUMBER	HTTP response code: HttpStatus.BAD_REQUEST Causes: Invalid number.
INVALID_OBJECT	HTTP response code: HttpStatus.BAD_REQUEST Causes: Invalid object.
INVALID_ PARAMETER	HTTP response code: HttpStatus.BAD_REQUEST Causes: API call/request used an invalid parameter, omitted a required parameter, or set the parameter to an incorrect value. Actions: • Review the errorDetails text to determine the invalid parameter that caused the error message. • Verify that your call/request included all required parameters. For example, for some content library API requests, the document path

Error Code	Details
	must begin with /contentlibrary and must contain the full document file name. • Verify correct syntax for the required parameters and their values. For example, you may receive this error if /contentlibrary is misspelled in a content library request, or if an incorrect file extension is given for a document file name.
INVALID_PROOF_ LAUNCH_TYPE	HTTP response code: HttpStatus.BAD_REQUEST Causes: Invalid Proof Launch Type.
INVALID_ REQUEST_ CONTENT	HTTP response code: HttpStatus.BAD_REQUEST Causes: Invalid request content.
INVALID_SESSION_ ID	HTTP response code: Causes: In SOAP, this can occur if something has happened to a valid server session state (the JSESSIONID cookie); for example, if may have been cleared or overwritten. Not applicable for REST.
INVALID_TOKEN	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Not a valid token. Actions: Ensure that the token matches that returned by Responsys during authentication. If you are unable to find the token returned by Responsys during authentication, you can also try to re-authenticate and use the newer token.
INVALID_USER_ NAME_PASSWORD	HTTP response code: HttpStatus.BAD_REQUEST Causes: This occurs during authentication if the API request

Error Code	Details
	uses an invalid user name or password, or attempts to log in to a suspended or deleted account. Actions: Contact the Responsys Account Administrator for assistance. The Responsys Account Administrator should go to the *Account | Manage Users* page in Responsys and take the following steps: • Verify that the API user name matches the one that the client application is using to authenticate. Responsys sends this error for deleted accounts, so you may not see the API user name in the list. • Verify that the API user account's User Status is set to Active. • If the password is the issue, ensure that the API user account's email is valid, and then reset the password. Responsys sends password reset emails to the user's email address. The API developer(s) should work with the person who receives the password reset email to get a secure password. Ensure that the team knows the Responsys account's password requirements. • If the API user exceed the number of unsuccessful login attempts (5 by default), Responsys locks the user login. Unlock the API user by selecting the user name from the list, and then clicking **Unlock Account**. Responsys sends an email to the API user, informing them that the login has been unlocked. The email recommends that the API user change the password immediately. • If single sign-on is enabled for the account, you will be unable to change the password. In this case, please contact Oracle Support.
LIMIT_NOT_VALID	HTTP response code: HttpStatus.BAD_REQUEST

Error Code	Details
	Causes: Limit is not valid.
LIST_ALREADY_ EXISTS	HTTP response code: HttpStatus.BAD_REQUEST Causes: List already exists.
LIST_NOT_FOUND	HTTP response code: HttpStatus.NOT_FOUND Causes: Not a valid client IP range.
LOGIN_BLOCKED_ INVALID_IPRANGE	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Login is blocked temporarily.
LOGIN_BLOCKED_ TEMPORARILY	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Login is blocked temporarily.
LOGINS_DISABLED	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Login is disabled.
MAX_ ATTACHMENT_ SIZE_EXCEEDED	HTTP response code: HttpStatus.BAD_REQUEST Causes: Attachment size exceeded.
MAX_LOGIN_ FAILURES_ EXCEEDED	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Maximum login failure exceeded.
MOBILE_ CAMPAIGN_ DISABLED_FOR_ USER	HTTP response code: HttpStatus.BAD_REQUEST Causes: Campaign disabled for user.
METHOD_NOT_ SUPPORTED	HTTP response code: HttpStatus.METHOD_NOT_ALLOWED Causes: Method not supported.
MULTIPLE_ OBJECTS_FOUND	HTTP response code: HttpStatus.BAD_REQUEST Causes: Multiple objects found.

Error Code	Details
MULTIPLE_ RECIPIENTS_ FOUND	HTTP response code: HttpStatus.BAD_REQUEST Causes: Multiple recipients found.
NO_CAMPAIGNS_ IN_THIS_FOLDER	HTTP response code: HttpStatus.BAD_REQUEST Causes: No campaigns for this folder.
NO_OBJECTS_IN_ THIS_FOLDER	HTTP response code: HttpStatus.BAD_REQUEST Causes: No objects in this folder.
NO_RECIPIENT_ FOUND	HTTP response code: HttpStatus.NOT_FOUND Causes: No recipient found.
OBJECT_ ALREADY_EXISTS	HTTP response code: HttpStatus.BAD_REQUEST Causes: Object already exists.
OBJECT_NOT_ FOUND	HTTP response code: HttpStatus.NOT_FOUND Causes: Object not found.
OFFSET_NOT_ VALID	HTTP response code: HttpStatus.BAD_REQUEST Causes: Offset is not valid.
OPERATION_NOT_ SUPPORTED	HTTP response code: HttpStatus.FORBIDDEN Causes: Operation not supported.
PASSWORD_ LOCKOUT	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Password locked.
PASSWORD_ EXPIRED	HTTP response code: HttpStatus.UNAUTHORIZED Causes: The API user's password has expired and must be reset.

Error Code	Details
	Actions: Contact the Responsys Account Administrator for assistance. The Responsys Account Administrator should go to the *Account | Manage Users* page in Responsys, ensure that the API user account's email is valid, and then reset the password. Responsys sends password reset emails to the user's email address. The API developer(s) should work with the person who receives the password reset email to get a secure password. Ensure that the team knows the Responsys account's password requirements.
PATH_NOT_VALID	HTTP response code: HttpStatus.BAD_REQUEST Causes: Object path not valid.
PRIVATE_KEY_ NOT_FOUND	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Private key not found.
PROFILE_LIST_ NOT_FOUND	HTTP response code: HttpStatus.NOT_FOUND Causes: Profile list not found.
PROFILE_LIST_ NOT_FOUND_IN_ FOLDER	HTTP response code: HttpStatus.NOT_FOUND Causes: Profile list not found.
PUSH_LIST_NOT_ FOUND	HTTP response code: HttpStatus.NOT_FOUND Causes: Push list not found.
RECIPIENT_LIMIT_ EXCEEDED	HTTP response code: HttpStatus.BAD_REQUEST Causes: Recipient limit exceeded.
RECIPIENT_ STATUS_ UNDELIVERABLE	HTTP response code: HttpStatus.OK

Error Code	Details
	Causes: Recipient status undeliverable.
RECORD_LIMIT_ EXCEEDED	HTTP response code: HttpStatus.BAD_REQUEST Causes: Record limit exceeded.
RECORD_NOT_ FOUND	HTTP response code: HttpStatus.NOT_FOUND Causes: Record not found.
RESOURCE_NOT_ FOUND	HTTP response code: HttpStatus.NOT_FOUND Causes: Resource not found.
SALESFORCE_ CAMPAIGN_ID_ NOT_FOUND	HTTP response code: HttpStatus.NOT_FOUND Causes: Salesforce campaign id not found.
SERVER_ CERTIFICATE_ EXPIRED	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Server certificate expired.
SERVER_ CERTIFICATE_ NOT_YET_VALID	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Server certificate not valid yet.
SERVER_ CERTIFICATE_ NOT_FOUND	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Server certificate not found.
SERVER_ CHALLENGES_DO_ NOT_MATCH	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Server challenge did not match.
SERVICE_ UNAVAILABLE	HTTP response code: HttpStatus.SERVICE_UNAVAILABLE Causes: Service unavailable.

Error Code	Details
TABLE_NOT_ FOUND	HTTP response code: HttpStatus.NOT_FOUND Causes: Table not found.
TOKEN_EXPIRED	HTTP response code: HttpStatus.UNAUTHORIZED Causes: In the REST API, this means that the authentication token has expired. Tokens expire after 2 hours, unless you use the call to refresh the token. Actions: • Log in again, and then use the new token in your call. Ensure that you also use the endpoint returned in your login call for subsequent calls. • Design your API code to account for token expiration. • You can design your API code to refresh your token if you do not want to re-authenticate completely every two hours.
UNABLE_TO_ CREATE_ CAMPAIGN	HTTP response code: HttpStatus.INTERNAL_SERVER_ ERROR Causes: Unable to create campaign. Actions: Contact the Responsys Account Administrator. The Account Administrator should verify the following: • The API user includes the correct role. • Whether the account is authorized to use the API.
UNEXPECTED_ EXCEPTION	HTTP response code: HttpStatus.INTERNAL_SERVER_ ERROR

Error Code	Details
	Causes: A server error occurred when processing your request. Actions: • Verify that the request body is correct for the endpoint; for example, this error may occur when using the parameters for a newer API request with an endpoint that expects parameters defined for the older version.
UNRECOVERABLE_ EXCEPTION	HTTP response code: HttpStatus.INTERNAL_SERVER_ ERROR Causes: Unrecoverable exception.
USER_BLOCKED	HTTP response code: HttpStatus.UNAUTHORIZED Cause: The API user has had 5 unsuccessful login attempts. Actions: Contact the Responsys Account Administrator for assistance. The Responsys Account Administrator should go to the *Account | Manage Users* page in Responsys and take the following steps • Unlock the API user by selecting the user name from the list, and then clicking **Unlock Account**. Responsys sends an email to the API user, informing them that the login has been unlocked. The email recommends that the API user change the password immediately. • Coordinate with the API user regarding the password reset. If the API user knows the old password, the API user can set a new one. Otherwise, you can reset the password from the *Responsys Account | Manage Users* page. Ensure that the API user has a valid

Error Code	Details
	email address, and then reset the password. API developers should work with the person who received the password reset email to ensure that the new password is secure.
VIRUS_FOUND	HTTP response code: HttpStatus.BAD_REQUEST Causes: Virus found in file. (NOTE: The `detail` property shows the name of the file in which the virus was found.) Actions: Follow your organization's processes to scan and repair files on the servers where the files are stored and where the client application is running.
WS_ARG_ DISABLED	HTTP response code: HttpStatus.UNAUTHORIZED Causes: Web service Account Resource Group (ARG) is disabled. Actions: Retry later. If the error message persists, create a My Oracle Support Service Request (SR).

How large can my request payload be, and what happens if it is too large?

Ensure that your request payload size is 10 MB or less. One way to reduce the request payload size is by referencing HTML content instead of sending it as part of the payload.

If your payload exceeds the maximum allowed size, you will receive a gateway server error. This error will not be in the standard Responsys error response format, because the request is not passed to Responsys by the gateway server. The gateway server will terminate the connection with the client application, and the client application will receive a `SocketException` error.

How can I find out the throttling limits for an API? I am getting API_LIMIT_EXCEEDED errors.

Use the Get Throttling Limits API. This REST request obtains a list of API throttling limits for key interfaces for your Responsys account. Note that it requires an authorization token, and it also uses a different endpoint path than the other REST API calls (`/rest/api/` instead of `/rest/api/v1.3/`).

How do I handle system outages?

Your client application must be designed to handle system outages, such as:

- **Scheduled outages**: Oracle Responsys undergoes maintenance downtimes on a monthly or bi-monthly schedule. During scheduled maintenance downtime, Oracle stops the Web Services server. Unless your client application is using an Automatic Failover for Transactional Messaging (AFTM) account, all Web Services API requests will fail during the maintenance period when your account's system is out of service.
- **Unscheduled outages**: All Responsys users receive an "Oracle Responsys Customer Bulletin" when there are system issues. These might be due to performance issues as well as unscheduled system outages.

During a system outage, attempts to make API calls will result in your receiving an HTTP response error similar to the following:

```
Remote Exception : Transport error: 503 Error: Service Temporarily
Unavailable
```

When it detects this error response, your client application code should take appropriate actions, which may include alerts to support staff, integration job queuing (because the Responsys system does not maintain a queue), and scheduled re-tries.

Where can I find system status information?

You can view system status for each system on Topliners at the following locations. You do not need to be a member of the Responsys Insiders group to view the system status information.

- Responsys Interact 2 System Status
- Responsys Interact 5 System Status

- Responsys SMS Customers System Status
- Responsys Push Customers System Status
- Responsys Legacy System (version 5) Customers System Status

How do I get help?

If you need help, create a My Oracle Support (MOS) service request at https://support.oracle.com. You can create service requests for permission requests, comments, or suggestions about Oracle Responsys documentation.

Within your service request, also known as an "SR", include the following details, as appropriate:

- Your pod and account name
- The endpoints in question for each call
- The request payloads that correspond with those endpoints
- The response from those calls, including any error messages and HTTP status codes

Migration notes

This topic provides information about the differences between REST API versions and how to best migrated your client application code to use the latest version.

- Version 1.1 to 1.3
- **Version 1.1 to 1.2**: Not applicable for the standard REST API. Version 1.2 applies only for accounts with Automatic Failover for Transactional Messaging (AFTM).
- Version 1 to 1.1

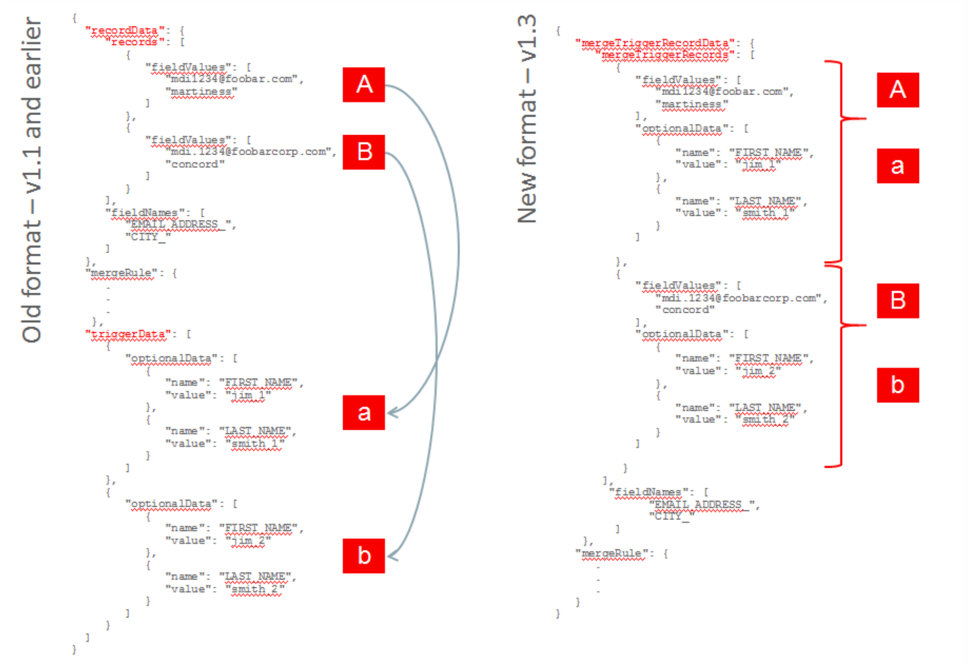
Version 1.1 to 1.3

This section contains information about the differences between the v1.1 and v1.3 API. The table that follows shows the API task affected, describes the old version functionality, and shows the new version's functionality.

API Task	Version 1.1 API	Version 1.3 API
Merge Trigger Email and Merge Trigger SMS	`records` array and `triggerData` array grouped the `field` values and `optional data` values separately, making it more difficult to ensure that the optional data was matched to the correct records, as shown in the image below (click image to enlarge).	New array, `mergeTriggerRecords`, now groups the field values and optional data in a pair-wise fashion, as shown in the image below (click image to enlarge).

API Task	Version 1.1 API	Version 1.3 API

NOTE: Using the new format with the v1.1 endpoint results in a 500 error code (Internal Server Error). Using the old format with the v1.3 endpoint results in a 400 error code (Bad Request).



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

```
{
   "mergeTriggerRecordData": {
      "mergeTriggerRecords": [
         {
            "fieldValues": [
               "mdi1234@foobar.com",
               "martiness"
            ],
            "optionalData": [
               {
                  "name": "FIRST_NAME",
                  "value": "jim_1"
               },
               {
                  "name": "LAST_NAME",
                  "value": "smith_1"
               }
            ]

         },
         {
            "fieldValues": [
               "mdi.1234@foobarcorp.com",
               "concord"
            ],
            "optionalData": [
               {
                  "name": "FIRST_NAME",
                  "value": "jim_2"
               },
               {
                  "name": "LAST_NAME",
                  "value": "smith_2"
               }
            ]

          }
      ],
       "fieldNames": [
             "EMAIL_ADDRESS_",
             "CITY_"
       ]
   },
  "mergeRule": {
     .
     .
     .
  }
}
```

Manage Email Campaign Schedule	Only supported using the campaign schedule calls for Email campaigns.	Now supports using the campaign schedule calls for Push campaigns. The account must have Integrated Push enabled. See the “Managing Campaign Schedules (Email or Push)” section for details.

Version 1 to 1.1

This section contains information about the differences between the v1 and v1.1 API.

API Task	Version 1 API	Version 1.1 API
Merge or update members in a profile extension table	`matchColumn` is deprecated **NOTE**: You may still see this in the response body of some v1.1 and greater APIs.	Use `matchColumnName1` and, optionally, `matchColumnName2`
Error Handling - For requests to trigger a message, when the response returns an error for a recipient then the success attribute is false.	The previous version of the API used to return the error description as part of the attribute `errorMessage`, with a description similar to the following: “Recipient deliverability status is undeliverable”.	The error description is now pre-pended with the error code, so that you can look up the error code from the `errorMessage` attribute: “*ERROR_CODE: error description*” For example: “RECIPIENT_ STATUS_ UNDELIVERABLE: Recipient deliverability status is undeliverable”.

Authentication with Certificates

The following example script illustrates the process for authentication with username and certificates.

```
private void loginWithCertificate() {

        String username = getUserInput("Username : ");
        byte[] clientChallengeBytes = String.valueOf(new Random
().nextLong()).getBytes();
        String encodedClientChallenge =
Base64.encodeBase64URLSafeString(clientChallengeBytes);

        MultiValueMap<String, String> requestParams = new
LinkedMultiValueMap<String, String>();
        requestParams.add("user_name", username);
        requestParams.add("auth_type", "server");
        requestParams.add("client_challenge",
encodedClientChallenge);

        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_FORM_
URLENCODED);

        HttpEntity<MultiValueMap<String, String>> requestEntity =
new HttpEntity<MultiValueMap<String, String>>(
                requestParams, headers);

        ServerChallengeResponse challengeResponse = null;
        try {
            challengeResponse = invokeApiWithPost(RestApi.LOGIN_
CERT_SVR, requestEntity,
                    ServerChallengeResponse.class);
        }
        catch (RestServiceException e) {
            sop("************************************");
            sop("Authenticate Server Failed");
            sop("************************************");
            sop("Error Code: " + e.getError().toString() +
"\r\nError: "
                    + e.getError().getErrorMessage() + "\r\nDetail
: " + e.getMessage());
            return;
        }
        byte[] encryptedClientChallengeBytes = Base64.decodeBase64
(challengeResponse
                .getClientChallenge());
        byte[] serverChallengeBytes = Base64.decodeBase64
(challengeResponse.getServerChallenge());

        String serverCertName = getUserInput("Enter the name &
location of the server certificate : ");
        File certFile = new File(serverCertName);
        if (!certFile.exists()) {
            sop("Server certificate doesn't exist in that
location");
            return;
        }

        X509Certificate serverCertificate = null;


        try {
            CertificateFactory certFactory =
CertificateFactory.getInstance("X.509");
            serverCertificate = (X509Certificate) certFactory
                    .generateCertificate(new FileInputStream
(certFile));

            Cipher decryptCipher = Cipher.getInstance("RSA");
            decryptCipher.init(Cipher.DECRYPT_MODE,
serverCertificate.getPublicKey());
            byte[] decryptedClientChallengeBytes = decryptCipher
                    .doFinal(encryptedClientChallengeBytes);

            // Compare the clientChallenge with
decryptedClientChallenge.
            boolean serverValidated = Arrays.equals
(clientChallengeBytes,
                    decryptedClientChallengeBytes);

            if (serverValidated) {
                sop("Server response validation 'PASSED' ...
proceeding further to login to REST service");
            }
            else {
                sop("Server response validation 'FAILED'");
                return;
            }
        }
        catch (Exception e) {
            sop("Could not process with the ceritificate : " +
e.getMessage());
            return;
        }
        // Get the private key of the client certificate
        PrivateKey privateKey = getPrivateKeyForClientCertificate
();
        if (privateKey == null) {
            sop("Couldn't get private key from the client
certificate");
            return;
        }

        byte[] encryptedServerChallengeBytes = null;

        try {
            Cipher encryptCipher = Cipher.getInstance("RSA");
            encryptCipher.init(Cipher.ENCRYPT_MODE, privateKey);
            encryptedServerChallengeBytes = encryptCipher.doFinal
(serverChallengeBytes);
        }
        catch (Exception e) {
            sop("Couldn't encrypt server challenge : " +
e.getMessage());
            return;
        }

        requestParams = new LinkedMultiValueMap<String, String>();
        requestParams.add("user_name", username);
        requestParams.add("auth_type", "client");
        requestParams.add("server_challenge",
                Base64.encodeBase64URLSafeString
(encryptedServerChallengeBytes));

        headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_FORM_
URLENCODED);
        headers.add("Authorization", challengeResponse.getAuthToken
());

        requestEntity = new HttpEntity<MultiValueMap<String,
String>>(requestParams, headers);

        LoginResponse loginResponse = null;
        try {
            loginResponse = invokeApiWithPost(RestApi.LOGIN_CERT_
CL, requestEntity,
                    LoginResponse.class);
        }
        catch (RestServiceException e) {
            sop("************************************");
            sop("Login with ceritficate : FAILED");
            sop("************************************");
            sop("Error Code: " + e.getError().toString() +
"\r\nError: "
                    + e.getError().getErrorMessage() + "\r\nDetail
: " + e.getMessage());
        }

        if (loginResponse != null) {
            String restEndPoint = loginResponse.getEndPoint();
            if (!isEmpty(restEndPoint)) {
                sop("************************************");
                sop("Login with ceritficate : PASSED");
                sop("************************************");
                this.svcEndPoint = restEndPoint;
                this.token = loginResponse.getAuthToken();
            }
            else {
                sop("************************************");
                sop("Login with user/pwd : FAILED");
                sop("************************************");
                sop("Error: RestEndPoint has not been defined for
this ARG."
                        + "Please check REST API settings in
PodConfig.ini.");
            }
        }
    }
```