



Oracle Responsys

28-Jul-2022

©2022 Oracle Corporation. All rights reserved

Asynchronous REST API Guide

REST API v1.3

Update 20A

NOTICE: The asynchronous Web Services APIs are currently released under our Controlled Availability program. To request access to this feature, please log in to [My Oracle Support](#) and create a service request. After Oracle enables your account for this feature, you can use the APIs described in this guide with Oracle Responsys 20A or later.

Information in this document is subject to change without notice. Data used as examples in this document is fictitious. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, without prior written permission of Oracle Responsys. Address permission requests, comments, or suggestions about Oracle Responsys documentation by creating a My Oracle Support (MOS) Service Request at <https://support.oracle.com>.

Contents

Overview	5
About this guide	5
Conventions used in this document	5
References	6
What's new to the Asynchronous REST API	8
Changes for 22B	8
Changes for 22A	9
Changes for 21D	9
Changes for 21C	9
Changes for 21B	10
Changes for 21A	10
How to Authenticate	12
Asynchronous APIs	14
Asynchronous Delete Content Library Items API	14
Asynchronous Delete Content Library Folders API	18
Asynchronous Delete Multiple Profile List Recipients API	22
Asynchronous Delete Supplemental Table Members API	26
Asynchronous Delete Multiple Profile Extension Recipients API	37
Asynchronous Merge Profile Extension Table Records API	41
Asynchronous Merge Profile List Members API	45
Asynchronous Merge Supplemental Table Records API	48
Asynchronous Merge Trigger Email API	52
Asynchronous Merge Trigger SMS API	56
Asynchronous Trigger a Custom Event	60

Asynchronous Trigger REI Event	64
Asynchronous Trigger Push Message API	68
Polling API endpoint	74
Workflow	74
Request details	75
Multiple Polling API endpoint	81
Request details	82
Returning reference data for asynchronous requests	85
Workflow	86
Asynchronous Reporting API	91
Create a request for Email campaign performance metrics	92
Create a request for SMS campaign performance metrics	94
Create a request for Mobile app campaign performance metrics	96
Polling API endpoint	98
Handling errors	105
Common error responses for the Asynchronous API	105
Learn more	107
Common error responses for the Polling API	107
Common error responses for the Reporting API	109

Overview

This document is for customers that use the Oracle Responsys Asynchronous REST API.

About this guide

The standard Oracle Responsys Web Services API processes API requests synchronously. For most tasks, Responsys sends a response shortly after the system finishes processing the call. This isn't always possible with database operations, where times to complete database operations are unpredictable. Delayed responses can lead to timeout issues when they run longer than your client application's timeout settings.

With the new Oracle Responsys Asynchronous REST API, we are introducing asynchronous processing for the Merge Trigger Email and Merge Trigger SMS tasks. Immediately after receiving an asynchronous API request, Responsys validates the payload and if the payload is well-formed JSON, the system returns an "ACCEPTED" status along with a transaction ID. You can poll later with a different API request to check the status of the original request corresponding to the transaction ID.

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

Convention	Meaning
	emphasis.
<i>italic</i>	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.

References

The standard Oracle Responsys REST API guides provide information that is also applicable to the asynchronous REST API, such as the Merge Trigger Email API request body and the standard API error responses and how to handle them. We provide direct links where applicable in this document, but for your reference, you can find the standard guides in the following locations:

- [Oracle Responsys REST API Developer's Guide](#) – A PDF version of our standard Responsys Web Services REST API Guide.

- *REST API for Oracle Responsys Marketing Cloud Service* – An HTML version of our standard Responsys Web Services REST API Guide. This guide contains the full request and response parameter descriptions for the API endpoints.

Before you begin

Before you use the asynchronous REST API, ensure that your Responsys account is properly configured by the Oracle Responsys Support team to have access to the Responsys Web Services Asynchronous REST API. This API is not standard and requires Oracle Responsys to enable it for your account.

What's new to the Asynchronous REST API

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

- [Update 22B](#)
- [Update 22A](#)
- [Update 21D](#)
- [Update 21C](#)
- [Update 21B](#)
- [Update 21A](#)

Changes for 22B

- We've added a new asynchronous API "Asynchronous Trigger REI Event". This endpoint enables you to asynchronously trigger a REI Event for Profile List, App Channel List, and Web Push recipients. [Learn more.](#)
- We've added a new asynchronous API "Asynchronous Delete Multiple Profile Extension Recipients API". This endpoint enables you to asynchronously delete members of an existing profile extension table. [Learn more.](#)
- The API limits for [Asynchronous Merge Trigger SMS API](#) have been enhanced. Please reach out to the support if you require a higher limit for this API.

Changes for 22A

- The [Responsys Asynchronous Reporting API](#) is now Generally Available to all customers. This includes access to all 4 API endpoints.
- The API limits for [Asynchronous Trigger SMS API](#) have been enhanced. Please reach out to the support if you require a higher limit for this API.

Changes for 21D

There are no significant updates to the Asynchronous REST API.

Changes for 21C

- We've added a new asynchronous API "Asynchronous Trigger a Custom Event". This endpoint enables you to asynchronously trigger a specific custom event. [Learn more.](#)
- We've added a new asynchronous API "Asynchronous Trigger Push Message". This endpoint enables you to asynchronously trigger a push message. [Learn more.](#)
- A new Multiple Polling API endpoint is now available. Poll up to 200 request IDs in a single request. [Learn more.](#)
- The Asynchronous Polling API now supports up to 10 request IDs per request. Combine multiple request IDs using the "&" operator. Specifying multiple request IDs enables you to retrieve more of your asynchronous requests in a single request. [Learn more.](#)

- We've added a new query parameter to the Asynchronous Polling API named `"?returnReferenceData=true"`. Enabling this parameter enables you to see any reference data specified in the original asynchronous request. Returning reference data is supported by the following endpoints: [Asynchronous Merge Trigger Email API](#), [Asynchronous Merge Trigger SMS API](#), and [Asynchronous Trigger Push Message API](#). [Learn more](#).
- We have significantly improved the Async API limits for the Polling API and Email Merge Trigger API. Access your latest limits using the [Get Throttling Limits API](#).

Changes for 21B

- The [Asynchronous REST API endpoints](#) are now Generally Available.

Changes for 21A

- We've added a new endpoint to enable you to delete content library items. See [Asynchronous Delete Content Library Items API](#) for details.
- We've added new metrics to the Reporting API. This new metrics are:
 - Responders (`responders`)
 - Average clicks (`averageClicks`)
 - Click per responder (`clickPerResponder`)
 - Responder rate (`responderRate`)
 - Unique clicks (`uniqueClicks`)

- Unique click-to-open rate (`uniqueClickToOpenRate`)

For more information, see the response body for the [Polling API endpoint](#).

How to Authenticate

The Asynchronous REST APIs use the same authentication endpoints as the standard REST APIs.

Before using the interfaces in this guide, ensure that your client application has authenticated with Responsys. Use the method described in either the “[Log in with username and password](#)” topic or the “[Log in with username and certificates](#)” topic of the document *REST API for Oracle Responsys Marketing Cloud Service*.

After your client application has authenticated successfully, it will receive the authentication response body shown below.

- When logging in with username and password, a successful response contains the values you need.
- Logging in with username and certificates involves two REST API requests, and the response containing the values you need is the second one received from the Responsys system (step 4 in the procedure mentioned above).

The response body for authentication contains the authToken and endPoint values:

```
{
  "authToken": " <auth_token> ",
  "issuedAt": <value>,
  "endPoint": " <endpoint_URI_value> "
}
```

Your client application must use the authToken values and endPoint values in all requests:

- Use the <endpoint_URI_value> when constructing the URI for your endpoint. The format may vary, depending on the programming language used to code the request. For example, if constructing a URL for the Merge Trigger Email request, it looks like this (endpoint URI + Service URL):

```
{endpoint_URI_value}/rest/asyncApi/v1.3/campaigns/{campaign_name}/email
```

- In the Request Header, substitute the response value for <auth_token> in places where we show Authorization: <auth_token>.

Asynchronous APIs

These are the Asynchronous APIs.

Note: The Asynchronous API is generally available for all Responsys customers.

Asynchronous Delete Content Library Items API

This asynchronous API enables you to delete multiple content library items in a single request (up to a maximum of 200 content library items). Use the [Polling API endpoint](#) to retrieve data about your request.

Note: The Asynchronous API is generally available for all Responsys customers.

Service URL:

`/rest/asyncApi/v1.3/clitems?action=delete`

Query Parameters:

action - Name of the action to perform. Must be set to delete.

Request Method:

POST

Request Header:

Authorization=<AUTH_TOKEN>

Content-Type=application/json

Request Body - Required Properties:

paths - A list of content library items to delete (200 maximum).

Sample Request Body:

```
{
  "paths": [
    "/contentlibrary/item.htm",
    "/contentlibrary/item1.htm"
  ]
}
```

Response Notes:

- The system validates the request payload syntax to ensure that the payload is well-formed JSON. However, it does not validate correctness of the values. For example, it does not validate if the campaign object exists or not, nor does it validate whether the payload uses the correct property names. Subsequently, you must poll for results to learn whether the request succeeded or failed due to other reasons.
- A successful response for the asynchronous API returns an HTTPS status code of 202 Accepted.
- The response body contains the information you need to analyze the response that you would get from the Polling API.
 - requestId: Unique identifier of the API request. The value is alphanumeric.
 - apiName: Name of the API request that you triggered. Always asyncMergeTriggerEmail for this API request's response.

- status: When successful, the status value is REQUEST_ACCEPTED. This means that the payload syntax was well-formed JSON and that the request was placed in the system queue for processing. You must poll for the final result; for more information, see [Polling API endpoint](#).
- See [Common error responses for the Asynchronous API](#) for details about the error responses.

Sample Successful Response:

```
{
  "requestId": "SXc4V092OjoxNjA2NzE4OTc4",
  "apiName": "asyncDeleteMultipleContentLibraryItems",
  "status": "REQUEST_ACCEPTED",
  "links": [
    {
      "rel": "self",
      "href": "/rest/asyncApi/v1.3/clItems?action=delete",
      "method": "POST"
    },
    {
      "rel": "asyncResponse",
      "href": "/rest/asyncApi/v1.3/requests/SXc4V092OjoxNjA2NzE4OTc4",
      "method": "GET"
    }
  ]
}
```

Sample Error Responses

Non content library items. Requests fail when non-content library items are passed in the request body. Error from the polling endpoint resembles:

```
{
  "response": {
    "paths": [
      "Path: /contentlibrary/folder/item1.htm does not exist.",
      "Path: /contentlibrary/folder/item2.htm does not exist.",
      "Path: /contentlibrary/folder/item3.htm does not exist."
    ]
  }
}
```

```

    ]
  },
  "requestId": "ZklQV2tHOjoxNjA2NzlwODM2",
  "apiName": "asyncDeleteMultipleContentLibraryItems",
  "timeZone": "India Standard Time",
  "requestTime": "2020-11-30T12:50:36.232",
  "requestProcessedTime": "2020-11-30T12:50:37.416",
  "status": "SUCCESS"
}

```

Duplicate content library items. Requests fail when duplicate content library items are passed in the request body. Error resembles:

```

"response": {
  "paths": [
    "/contentlibrary/item1.htm",
    "Duplicate content library item path [/contentlibrary/item1.htm].",
    "Path: /contentlibrary/item2.htm does not exist.",
    "/contentlibrary/item2.htm",
    "Duplicate content library item path [/contentlibrary/item2.htm].",
    "Path: /contentlibrary/item2.htm does not exist."
  ]
}

```

Limit exceeded. The maximum number of content library items that can be deleted per request is 200. If more than 200 items are specified in one request, the following error appears:

```

{
  "type": "",
  "title": "Record limit exceeded",
  "errorCode": "RECORD_LIMIT_EXCEEDED",
  "detail": "Maximum 200 Items can be deleted.",
  "errorDetails": []
}

```

Asynchronous Delete Content Library Folders API

This asynchronous API enables you to delete multiple content library folders in a single request. Use the [Polling API endpoint](#) to retrieve data about your request.

Note: The Asynchronous API is generally available for all Responsys customers.

Service URL:

`/rest/asyncApi/v1.3/clFolders?action=delete`

Query Parameters:

action - Name of the action to perform. Must be set to delete.

Request Method:

POST

Request Header:

Authorization=<AUTH_TOKEN>

Content-Type=application/json

Request Body - Required Properties:

paths - A list of content library folders to delete.

Sample Request Body:

```
{
  "paths": [
    "/contentlibrary/folder",
    "/contentlibrary/folder1"
  ]
}
```

Response Notes:

- The system validates the request payload syntax to ensure that the payload is well-formed JSON. However, it does not validate correctness of the values. For example, it does not validate if the campaign object exists or not, nor does it validate whether the payload uses the correct property names. Subsequently, you must poll for results to learn whether the request succeeded or failed due to other reasons.
- A successful response for the asynchronous API returns an HTTPS status code of 202 Accepted.
- The response body contains the information you need to analyze the response that you would get from the Polling API.
 - requestId: Unique identifier of the API request. The value is alphanumeric.
 - apiName: Name of the API request that you triggered. Always asyncMergeTriggerEmail for this API request's response.
 - status: When successful, the status value is REQUEST_ACCEPTED. This means that the payload syntax was well-formed JSON and that the request was placed in the system queue for processing. You must poll for the final result; for more information, see [Polling API endpoint](#).
- See [Common error responses for the Asynchronous API](#) for details about the error responses.

Sample Successful Response:

```
{
  "requestId": "OUZkeTFIOjoxNTY1NzAxNjY0",
  "apiName": "asyncDeleteMultipleContentLibraryFolders",
  "status": "REQUEST_ACCEPTED",
  "links": [
    {
      "rel": "self",
      "href": "/rest/asyncApi/v1.3/clFolders?action=delete",
      "method": "POST"
    },
    {
      "rel": "asyncResponse",
      "href": "/rest/asyncApi/v1.3/requests/OUZkeTFIOjoxNTY1NzAxNjY0",
      "method": "GET"
    }
  ]
}
```

Sample Error Responses

Empty list. Requests fail when there is an empty list in the request body. Error resembles:

```
{
  "type": "",
  "title": "Invalid request content",
  "errorCode": "INVALID_REQUEST_CONTENT",
  "detail": "Folder paths should not be empty. ",
  "errorDetails": []
}
```

Limit exceeded. The maximum number of content library folders that can be deleted per request is 200. If more than 200 folders are specified in one request, the following error appears:

```
{
  "type": "",
  "title": "Record limit exceeded",
  "errorCode": "RECORD_LIMIT_EXCEEDED",
  "detail": "Maximum 200 folders can be deleted.",
  "errorDetails": []
}
```

Folder not found. The following example shows a sample error response for the [Polling API endpoint](#) – in this case, the paths object contains the error JSON, which tells us that the content library folders we tried to delete, did not exist. A response code of 200 OK is shown, however, because the polling request itself completed successfully.

```
{
  "response": {
    "paths": [
      "Deletion Failed: /contentlibrary/path1. FolderFault|FOLDER_NOT_FOUND|/contentlibrary/path1",
      "Deletion Failed: /contentlibrary/clPath2. FolderFault|FOLDER_NOT_FOUND|/contentlibrary/clPath2"
    ]
  },
  "requestId": "OUZkeTFIOjoxNTY1NzAxNjY0",
  "apiName": "asyncDeleteMultipleContentLibraryFolders",
  "timeZone": "India Standard Time",
  "requestTime": "2019-08-13T18:37:44.673",
  "requestProcessedTime": "2019-08-13T18:37:45.517",
  "status": "SUCCESS"
}
```

Duplicate folders. The following example shows a sample error response for the [Polling API endpoint](#) – in this case, the request body contains duplicate folders. A

response code of 200 OK is shown, however, because the polling request itself completed successfully.

```
{
  "response": {
    "paths": [
      "contentlibrary/cl1",
      "Duplicate folder path [contentlibrary/cl1].",
      "Duplicate folder path [contentlibrary/cl1]."
    ]
  },
  "requestId": "UFBjUERN0joxNTU2MjY5MDM2",
  "apiName": "asyncDeleteMultipleContentLibraryFolders",
  "timeZone": "India Standard Time",
  "requestTime": "2019-04-26T14:27:16.377",
  "requestProcessedTime": "2019-04-26T14:27:16.917",
  "status": "SUCCESS"
}
```

Asynchronous Delete Multiple Profile List Recipients API

This asynchronous API enables you to delete multiple profile list recipients in a single request. Use the [Polling API endpoint](#) to retrieve data about your request.

Service URL:

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

Required Path Parameters:

- `listName` - Name of the list.

Query Parameters:

- `action` - Name of the action to perform. Must be set to `delete`.

Request Method:

POST

Sample Request Body:

```
{
  "queryAttribute": "e",
  "ids": [
    "recipient1@example.com",
    "jnellore-83@interactexpress.com"
  ]
}
```

Response Notes:

- A successful response for the asynchronous API returns an HTTPS status code of 202 Accepted.
- A successful response returns a requestId, to be used in the [Polling API endpoint](#) to retrieve the status of your request.
- See [Common error responses for the Polling API](#) for details about the error responses.

Sample Successful Response:

```
{
  "requestId": "NU8xM284OjoxNTgyNTI3NzQ5",
  "apiName": "asyncDeleteMultipleListRecipients",
  "status": "REQUEST_ACCEPTED",
  "links": [
    {
      "rel": "self",
      "href": "/rest/asyncApi/v1.3/lists/16K_REC/members?action=delete",
      "method": "POST"
    },
    {
      "rel": "asyncResponse",
      "href": "/rest/asyncApi/v1.3/requests/NU8xM284OjoxNTgyNTI3NzQ5",
    }
  ]
}
```

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

Sample Successful Response from the Polling API:

```
{
  "response": {
    "recordData": {
      "records": [
        [
          "DELETEFAILED: Record not Found in the List."
        ],
        [
          "jnellore-83@interactexpress.com"
        ]
      ],
      "mapTemplateName": null,
      "fieldNames": [
        "EMAIL_ADDRESS_"
      ]
    }
  },
  "requestId": "a2RmZDlyOjoxNTgyNTI4MTIx",
  "apiName": "asyncDeleteMultipleListRecipients",
  "timeZone": "India Standard Time",
  "requestTime": "2020-02-24T12:38:41.153",
  "requestProcessedTime": "2020-02-24T12:38:41.858",
  "status": "SUCCESS"
}
```

Sample Error Messages

Invalid query attribute. The following example shows a sample error response when a request parameter contains invalid characters. The query attribute must be either r, e, c, or m. Error resembles:

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

Invalid list. Requests fail when there is an invalid list in the request body. Error resembles:

```
{
  "response": {
    "errorCode": "LIST_NOT_FOUND",
    "detail": "16K_REC11 List Not Found",
    "type": "",
    "title": "List not found",
    "errorDetails": "[]"
  },
  "requestId": "QWZ3ajV6OjoxNTgyNTI5NTk0",
  "apiName": "asyncDeleteMultipleListRecipients",
  "timeZone": "India Standard Time",
  "requestTime": "2020-02-24T13:03:14.899",
  "requestProcessedTime": "2020-02-24T13:03:15.070",
  "status": "ERROR"
}
```

Limit Exceeded. The maximum number of records allowed per API call is 200. If more than 200 items are specified in one request, the following error appears:

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

```
"errorDetails": [  
]  
}
```

Learn more

Asynchronous APIs

Asynchronous Delete Supplemental Table Members API

This asynchronous API enables you to delete multiple supplemental table members in a single request. Use the [Polling API endpoint](#) to retrieve data about your request.

Note: The Asynchronous API is generally available for all Responsys customers.

Service URL:

/rest/asyncApi/v1.3/folders/{folderName}/suppData/
{tableName}/members/members?action=delete

Required Path Parameters:

- **folderName** - Name of the folder where the supplemental table is found.
- **tableName** - Name of the supplemental table.

Query Parameters:

- `action` - Name of the action to perform. Must be set to delete.

Request Method:

POST

Request Header:

Authorization=<AUTH_TOKEN>

Content-Type=application/json

Request Body - Required Properties:

- `recordData` - Record data that represents field names and corresponding record values for the members.
 - `fieldNames` - Name of the primary key to match supplemental table members.
 - `records` - Values corresponding to the `fieldNames` specified.

Sample Request Body:

```
{
  "recordData": {
    "fieldNames": [
      "CUST_NO",
      "EMAIL_ADDRESS"
    ],
    "records": [
      [
        "100",
        "api.user100@oracle.com"
      ],
      [
        "200",
```

```
"api.user200@oracle.com"
],
[
  "300"
]
]
}
}
```

Response Notes:

- The system validates the request payload syntax to ensure that the payload is well-formed JSON. However, it does not validate correctness of the values. For example, it does not validate if the campaign object exists or not, nor does it validate whether the payload uses the correct property names. Subsequently, you must poll for results to learn whether the request succeeded or failed due to other reasons.
- A successful response for the asynchronous API returns an HTTPS status code of 202 Accepted.
- The response body contains the information you need to analyze the response that you would get from the Polling API.
 - requestId: Unique identifier of the API request. The value is alphanumeric.
 - apiName: Name of the API request that you triggered. Always asyncMultipleDeleteTableRecords for this API request's response.
 - status: When successful, the status value is REQUEST_ACCEPTED. This means that the payload syntax was well-formed JSON and that the request was placed in the system queue for processing. You must poll for the final result; for more information, see [Polling API endpoint](#).
- See [Common error responses for the Asynchronous API](#) for details about the error responses.

Sample Successful Response:

```
{
  "requestId": "OUZkeTFIOjoxNTY1NzAxNjY0",
  "apiName": "asyncMultipleDeleteTableRecords",
  "status": "REQUEST_ACCEPTED",
  "links": [
    {
      "rel": "self",
      "href": "/rest/asyncApi/v1.3/folders/{folderName}/suppData/{tableName}/members/members?action=delete",
      "method": "POST"
    },
    {
      "rel": "asyncResponse",
      "href": "/rest/asyncApi/v1.3/requests/OUZkeTFIOjoxNTY1NzAxNjY0",
      "method": "GET"
    }
  ]
}
```

Sample Successful Response from the Polling API:

```
{
  "response": {
    "recordData": {
      "fieldNames": [
        "CUST_NO",
        "EMAIL_ADDRESS"
      ],
      "records": [
        [
          {
            "ids": [
              "118",
              "api.user@oracle.com"
            ],
            "success": true
          }
        ]
      ]
    }
  }
}
```

```

    ]
  ]
}
},
"requestId": "WmVFSGZqOjoxNjE0NjA0MTUx",
"apiName": "asyncMultipleDeleteTableRecords",
"timeZone": "Pacific Standard Time",
"requestTime": "2021-03-01T05:09:11.192",
"requestProcessedTime": "2021-03-01T05:09:11.493",
"status": "SUCCESS"
}

```

Sample Error Responses

Folder name invalid or not found. The following example shows a sample error response when the folder name specified is invalid or not found. The response is a 404 Not Found error.

```

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

```

Supplemental table not found. The following example shows a sample error response when the supplemental table name specified is invalid or not found. The response is a 404 Not Found error.

```

{
  "type": "",
  "title": "Table not found",
  "errorCode": "TABLE_NOT_FOUND",
  "detail": "SupplementalTable1 Table Not Found",
}

```

```
"errorDetails": []  
}
```

Invalid supplemental table name. The following example shows a sample error response when the supplemental table name specified contains invalid characters. The response is a 400 Bad Request error.

```
{  
  "type": "",  
  "title": "Invalid request parameters",  
  "errorCode": "INVALID_PARAMETER",  
  "detail": "Invalid characters in the Table Name",  
  "errorDetails": []  
}
```

Supplemental table name exceeds 100 characters. The following example shows a sample error response when the supplemental table name specified exceeds 100 characters. The response is a 400 Bad Request error.

```
{  
  "type": "",  
  "title": "Invalid request parameters",  
  "errorCode": "INVALID_PARAMETER",  
  "detail": "Invalid characters in the Table Name",  
  "errorDetails": []  
}
```

No matching records found. The following example shows a sample error response for the [Polling API endpoint](#) – in this case, the `records` object contains the error JSON, which tells us that the records we tried to delete, did not exist. A response code of 200 OK is shown, however, because the polling request itself completed successfully.

```

{
  "response": {
    "recordData": {
      "fieldNames": [
        "EMAIL_ADDRESS_"
      ],
      "records": [
        [
          {
            "ids": [
              "api.user@oracle.com"
            ],
            "success": false,
            "errorCode": "RECORD_NOT_FOUND",
            "errorMessage": "No records found in the table for the given ids"
          }
        ],
        [
          {
            "ids": [
              "api.user2@oracle.com"
            ],
            "success": false,
            "errorCode": "RECORD_NOT_FOUND",
            "errorMessage": "No records found in the table for the given ids"
          }
        ]
      ]
    }
  },
  "requestId": "M0d0S2RrOjoxNjE0ODM0NTM1",
  "apiName": "asyncMultipleDeleteTableRecords",
  "timeZone": "India Standard Time",
  "requestTime": "2021-03-04T10:38:55.744",
  "requestProcessedTime": "2021-03-04T10:38:56.066",
  "status": "SUCCESS"
}

```

Invalid field name. The following example shows a sample error response when the specified field (EMAIL_ADDRESS) is not found in the supplemental table.

```
{
  "response": {
    "errorCode": "INVALID_FIELD_NAME",
    "detail": "Column(s) [EMAIL_ADDRESS] not found in the table",
    "type": "",
    "title": "Invalid field name",
    "errorDetails": "[]"
  },
  "requestId": "ekITb2o0OjoxNjE0MDgwMzE5",
  "apiName": "asyncMultipleDeleteTableRecords",
  "timeZone": "Pacific Standard Time",
  "requestTime": "2021-02-23T03:38:39.219",
  "requestProcessedTime": "2021-02-23T03:38:39.657",
  "status": "ERROR"
}
```

Invalid field name. The following example shows a sample error response when not all the primary keys of a supplemental table are specified.

```
{
  "response": {
    "errorCode": "INVALID_PARAMETER",
    "detail": "All and Only the Primary Keys in the Table [EMAIL_ADDRESS_, INTEGER_FIELD, NUMBER_FIELD, FIELD_DELETE] must be specified as Query Columns.",
    "type": "",
    "title": "Invalid request parameters",
    "errorDetails": "[]"
  },
  "requestId": "UnNLY3IWOjoxNjE0NTM0MTEy",
  "apiName": "asyncMultipleDeleteTableRecords",
  "timeZone": "India Standard Time",
  "requestTime": "2021-02-28T23:11:52.993",
  "requestProcessedTime": "2021-02-28T23:11:53.240",
  "status": "ERROR"
}
```

```
}
```

Record limit exceeded. The following example shows a sample error response when more than 200 records are specified in the request. The response is a 400 Bad Request error.

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

Records is null or empty. The following example shows a sample error response when no field values are specified in the request payload. The response is a 400 Bad Request error.

```
{
  "type": "",
  "title": "Invalid request parameters",
  "errorCode": "INVALID_PARAMETER",
  "detail": "Record(s) is null or empty ",
  "errorDetails": []
}
```

Empty or null passed for field values. The following example shows a sample error response when null or empty field values are specified in the request payload. The response is a 400 Bad Request error.

```
{
  "response": {
```

```

"recordData": {
  "fieldNames": [
    "EMAIL_ADDRESS_"
  ],
  "records": [
    [
      {
        "Ids": [
          ""
        ],
        "success": false,
        "errorCode": "INVALID_PARAMETER",
        "errorMessage": "ERROR: Column EMAIL_ADDRESS_ value cannot be empty\n\n"
      }
    ],
    [
      {
        "Ids": [
          null
        ],
        "success": false,
        "errorCode": "INVALID_PARAMETER",
        "errorMessage": "ERROR: Column EMAIL_ADDRESS_ value cannot be empty\n\n"
      }
    ]
  ]
},
"requestId": "RmJCTWhEOjoxNjE0ODM0NzQz",
"apiName": "asyncMultipleDeleteTableRecords",
"timeZone": "India Standard Time",
"requestTime": "2021-03-04T10:42:23.925",
"requestProcessedTime": "2021-03-04T10:42:24.239",
"status": "SUCCESS"
}

```

Field names are null or empty. The following example shows a sample error response when field names are either null or empty.

```
{
  "response": {
    "errorCode": "INVALID_FIELD_NAME",
    "detail": "One of the field names is null OR empty",
    "type": "",
    "title": "Invalid field name",
    "errorDetails": "[]"
  },
  "requestId": "NU1JbUM1OjoxNjE0MDgwMzY0",
  "apiName": "asyncMultipleDeleteTableRecords",
  "timeZone": "Pacific Standard Time",
  "requestTime": "2021-02-23T03:39:24.570",
  "requestProcessedTime": "2021-02-23T03:39:24.632",
  "status": "ERROR"
}
```

Not all primary keys have been specified. The following example shows a sample error response.

```
{
  "response": {
    "errorCode": "INVALID_PARAMETER",
    "detail": "All and Only the Primary Keys in the Table [EMAIL_ADDRESS, CUST_NO] must be specified as Query Columns.",
    "type": "",
    "title": "Invalid request parameters",
    "errorDetails": "[]"
  },
  "requestId": "SWxsRkplOjoxNjE0MDgwNDE2",
  "apiName": "asyncMultipleDeleteTableRecords",
  "timeZone": "Pacific Standard Time",
  "requestTime": "2021-02-23T03:40:16.371",
  "requestProcessedTime": "2021-02-23T03:40:16.404",
  "status": "ERROR"
}
```

Non-primary key specified. The following example shows a sample error response.

```
{
  "response": {
    "errorCode": "INVALID_FIELD_NAME",
    "detail": "Column(s) [FIRST_NAME] is not Primary Key(s)",
    "type": "",
    "title": "Invalid field name",
    "errorDetails": "[]"
  },
  "requestId": "Sm5ORFBTOjoxNjE0MDgwOTM3",
  "apiName": "asyncMultipleDeleteTableRecords",
  "timeZone": "Pacific Standard Time",
  "requestTime": "2021-02-23T03:48:57.764",
  "requestProcessedTime": "2021-02-23T03:48:57.825",
  "status": "ERROR"
}
```

Asynchronous Delete Multiple Profile Extension Recipients API

This endpoint is used to delete members of an existing profile extension table. For a given profile extension table, an array of record data that contain field names and their corresponding field values are specified. This endpoint is the asynchronous alternative to [Delete Multiple Profile Extension Recipients](#).

A maximum of 200 records can be merged in one request.

Note: The Asynchronous API is generally available for all Responsys customers.

Service URL:

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

Required Path Parameters:

listName - Name of the profile extension table's parent profile list.

petName - name of the profile extension table.

Request Method:

POST

Request Header:

Authorization=<AUTH_TOKEN>

Content-Type=application/json

Request Body - Required Properties:

- **ids** – Record data that represents field names and corresponding values for the member.
- **queryAttribute** - Query Attribute based on which records will be deleted. Can have the values:
 - **r** (RIID)
 - **e** (email address)
 - **m** (mobile number)
 - **c** (customer id)

For more information about forming a request, see the [Delete multiple Profile Extension Recipients](#) in the Oracle Responsys REST API Guide. View the Request section of the topic.

Sample Request Body:

```
{
  "queryAttribute": "e",
  "ids": [
    "john.doe@oracle.com",
    "adam.silver@oracle.com",
    "john.appleseed@oracle.com"
  ]
}
```

Response Notes:

- A successful response for the asynchronous API returns an HTTPS status code of 202 Accepted.
- A successful response returns a requestId, to be used in the [Polling API endpoint](#) to retrieve the status of your request.
- See [Common error responses for the Asynchronous API](#) for details about the error responses.

Sample Response:

```
{
  "requestId": "c3RuMGhIOjoxNjQzNzA4NjYz",
  "apiName": "asyncDeleteMultipleProfileExtensionRecipients",
  "status": "REQUEST_ACCEPTED",
  "links": [
    {
      "rel": "self",
      "href": "/rest/asyncApi/v1.3/lists/Summer_Event_Attendees/listExtensions/PET_records_Summer_Event/members?action=delete",
    }
  ]
}
```

```
    "method": "POST"
  },
  {
    "rel": "asyncResponse",
    "href": "/rest/asyncApi/v1.3/requests/c3RuMGhIOjoxNjQzNzA4NjYz",
    "method": "GET"
  }
]
}
```

Sample successful response from Polling API:

```
{
  "response": {
    "recordData": {
      "records": [
        [
          "john.doe@oracle.com"
        ],
        [
          "adam.silver@oracle.com"
        ],
        [
          "john.appleseed@oracle.com"
        ]
      ],
      "mapTemplateName": null,
      "fieldNames": [
        "EMAIL_ADDRESS_"
      ]
    }
  },
  "requestId": "c3RuMGhIOjoxNjQzNzA4NjYz",
  "apiName": "asyncDeleteMultipleProfileExtensionRecipients",
  "timeZone": "Pacific Standard Time",
  "requestTime": "2022-02-01T01:44:23.032",
  "requestProcessedTime": "2022-02-01T01:44:23.354",
  "status": "SUCCESS"
}
```

Asynchronous Merge Profile Extension Table Records API

This endpoint is used to add new members to an existing profile extension table, or update data for existing members within the table. For a given profile extension table, an array of record data that contain field names and their corresponding field values are specified. This endpoint is the asynchronous alternative to [Merge Profile Extension Recipients](#).

A maximum of 200 records can be merged in one request.

Note: The Asynchronous API is generally available for all Responsys customers.

Service URL:

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

Required Path Parameters:

listName - Name of the profile extension table's parent profile list.

petName - name of the profile extension table.

Request Method:

POST

Request Header:

Authorization=<AUTH_TOKEN>

Content-Type=application/json

Request Body - Required Properties:

`recordData` – Record data that represents field names and corresponding values for the member.

For more information about forming a request, see the [Merge Profile Extension Recipients REST Endpoints](#) page in the *Oracle Responsys REST API Guide*. View the Request section of the topic.

Request Body - Optional Properties:

`insertOnNoMatch` – Boolean value that specifies whether a member record should be inserted into the Supplemental Table if it does not already exist.

`updateOnMatch` – Specifies whether a member record should be updated in the Supplemental Table if it already exists. Allowed values: REPLACE_ALL and NO_UPDATE.

`matchColumnName1` and `matchColumnName2` – The column name to be used to match the recipient record to the Profile Extension records. Only used during the merge operation. Allowed values: RIID, CUSTOMER_ID, EMAIL_ADDRESS, MOBILE_NUMBER, EMAIL_MD5_HASH, EMAIL_SHA256_HASH.

For more information about forming a request, see the [Merge Profile Extension Recipients REST Endpoints](#) page in the *Oracle Responsys REST API Guide*. View the Request section of the topic.

Sample Request Body:

```
{
  "recordData": {
    "fieldNames": [
      "EMAIL_ADDRESS_",
      "EMPID",
      "MOBILE_NUMBER_",
      "EMAIL_SHA256_HASH_"
    ],
    "records": [
      [
        "email.address@oracle.com",
        "09",
        "0123456789",
        "456f90d1f6a99f605043fdc5ec1384e1213a718e8259e7676e8e7ffb7fad0d2d"
      ],
      [
        "email.address1@oracle.com",
        "09",
        "9845098450",
        "456f90d1f6a99f605043fdc5ec1384e1213a718e8259e7676e8e7ffb7fad0d2d"
      ],
      [
        "email.address2@oracle.com",
        "09",
        "2598233819",
        "456f90d1f6a99f605043fdc5ec1384e1213a718e8259e7676e8e7ffb7fad0d2d"
      ]
    ],
    "mapTemplateName": ""
  },
  "insertOnNoMatch": true,
  "updateOnMatch": "REPLACE_ALL",
  "matchColumnName1": "EMAIL_SHA256_HASH",
  "matchColumnName2": ""
}
```

For more information, see the [Merge Profile Extension Recipients REST Endpoints page](#) in the *Oracle Responsys REST API Guide*. View the Example section of the topic.

Response Notes:

- A successful response for the asynchronous API returns an HTTPS status code of 202 Accepted.
- A successful response returns a requestId, to be used in the [Polling API endpoint](#) to retrieve the status of your request.
- See [Common error responses for the Asynchronous API](#) for details about the error responses.

Sample Response:

```
{
  "requestId": "eVAyTWJGOjoxNTc2MDkzMTI2",
  "apiName": "asyncMergeIntoProfileExtension",
  "status": "REQUEST_ACCEPTED",
  "links": [
    {
      "rel": "self",
      "href": "/rest/asyncApi/v1.3/lists/My_List/listExtensions/Pet_Name/members",
      "method": "POST"
    },
    {
      "rel": "asyncResponse",
      "href": "/rest/asyncApi/v1.3/requests/eVAyTWJGOjoxNTc2MDkzMTI2",
      "method": "GET"
    }
  ]
}
```

Asynchronous Merge Profile List Members API

This endpoint is used to add new members to an existing profile list, or update data for existing members within the list. For a given list in a specific folder, an array of record data that can contain field names and their corresponding field values are specified. This endpoint is the asynchronous alternative to [Merge List Recipients](#).

A maximum of 200 records can be merged in one request.

Note: The Asynchronous API is generally available for all Responsys customers.

Service URL:

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

Required Path Parameters:

`listName` - Name of the list.

Request Method:

POST

Request Header:

Authorization=<AUTH_TOKEN>

Content-Type=application/json

Request Body - Required Properties:

recordData – Record data that represents field names and corresponding values for the member.

mergeRule – Merge rule used to merge recipient records in a Profile List. Only used during the merge operation.

For more information about forming a request, see the [Merge List Recipients REST Endpoints page](#) in the *Oracle Responsys REST API Guide*. View the Request section of the topic.

Sample Request Body:

```
{
  "recordData": {
    "fieldNames": [
      "riid_",
      "mobile_number_",
      "email_address_"
    ],
    "records": [
      [
        "4094326",
        "9845349498",
        "ab.cd@gmail.com"
      ],
      [
        "4094327",
        "9844444444",
        "unknown@oracle.com"
      ],
      [
        "4094328",
        "98444444666",
        "abc@gmail.com"
      ]
    ]
  }
}
```

```

    ],
    [
      "ssdcf",
      "98444444444",
      "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"
}
}

```

Response Notes:

- A successful response for the asynchronous API returns an HTTPS status code of 202 Accepted.
- A successful response returns a requestId, to be used in the [Polling API endpoint](#) to retrieve the status of your request.
- See [Common error responses for the Asynchronous API](#) for details about the error responses.

Sample Response:

```
{
  "requestId": " cDBtY3lFOjoxNTc2MDkzMzk3",
  "apiName": "asyncMergeListMembers",
  "status": "REQUEST_ACCEPTED",
  "links": [
    {
      "rel": "self",
      "href": "/rest/asyncApi/v1.3/lists/My_List_Name/members",
      "method": "POST"
    },
    {
      "rel": "asyncResponse",
      "href": "/rest/asyncApi/v1.3/requests/cDBtY3lFOjoxNTc2MDkzMzk3",
      "method": "GET"
    }
  ]
}
```

Asynchronous Merge Supplemental Table Records API

This endpoint is used to add new members to an existing supplemental table, or update data for existing members within the table. For a given supplemental table in a specific folder, an array of record data that contains field names and their corresponding field values are specified. This endpoint is the asynchronous alternative to [Merge Table Records](#).

A maximum of 200 records can be merged in one request.

Note: The Asynchronous API is generally available for all Responsys customers.

Service URL:

/rest/asyncApi/v1.3/folders/{folderName}/suppData/{suppTable}/members

Required Path Parameters:

folderName - Name of the folder where the supplemental table is found.

suppTable - Name of the supplemental table.

Request Method:

POST

Request Header:

Authorization=<AUTH_TOKEN>

Content-Type=application/json

Request Body - Required Properties:

recordData – Record data that represents field names and corresponding values for the member.

For more information about forming a request, see the [Merge Table Records REST Endpoints](#) page in the *Oracle Responsys REST API Guide*. View the Request section of the topic.

Request Body - Optional Properties

- `insertOnNoMatch` – Boolean value that specifies whether a member record should be inserted into the Supplemental Table if it does not already exist.
- `updateOnMatch` – Specifies whether a member record should be updated in the Supplemental Table if it already exists. Allowed values: `REPLACE_ALL` and `NO_UPDATE`.

For more information about forming a request, see the [Merge Table Records REST Endpoints](#).

Sample Request Body:

```
{
  "recordData": {
    "fieldNames": [
      "EMAIL_ADDRESS_",
      "INTEGER_FIELD"
    ],
    "records": [
      [
        "email.address@oracle.com",
        "10"
      ],
      [
        "email.address@oracle.com",
        "11"
      ],
      [
        "email.address@oracle.com",
        "12"
      ],
      [
        "email.address@oracle.com",
        "13"
      ]
    ]
  }
}
```

```

    "mapTemplateName": "Map_Template_Name"
  },
  "insertOnNoMatch": true,
  "updateOnMatch": "REPLACE_ALL"
}

```

Response Notes:

- A successful response for the asynchronous API returns an HTTPS status code of 202 Accepted.
- A successful response returns a requestId, to be used in the [Polling API endpoint](#) to retrieve the status of your request.
- See [Common error responses for the Asynchronous API](#) for details about the error responses.

Sample Response:

```

{
  "requestId": "TzJGZ0w50joxNTc2MDg1Mzk4",
  "apiName": "asyncMergeTableRecords",
  "status": "REQUEST_ACCEPTED",
  "links": [
    {
      "rel": "self",
      "href": "/rest/asyncApi/v1.3/folders/My_Folder_Name/suppData/My_Table_Name/members",
      "method": "POST"
    },
    {
      "rel": "asyncResponse",
      "href": "/rest/asyncApi/v1.3/requests/TzJGZ0w50joxNTc2MDg1Mzk4",
      "method": "GET"
    }
  ]
}

```

Asynchronous Merge Trigger Email API

The Asynchronous Merge Trigger Email API has a different service URL than the standard Merge Trigger Email API, but it uses the same parameters, header, and request body. The response is different: it provides the data necessary to use the [Polling API endpoint](#) results.

Note: The Asynchronous API is generally available for all Responsys customers.

Service URL:

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

Required Path Parameters:

`campaignName` - Name of the email campaign to send to the recipients

Request Method:

POST

Request Header:

Authorization=<AUTH_TOKEN>

Content-Type=application/json

Message-Expiration=<ttl value – if omitted, the system uses the default of 3600 seconds. Use this value to set an expiration time for your request.>

Request Body Notes:

- You can learn about the required parameters in the [Merge Trigger Email REST Endpoints page](#). View the Request section of the topic.
- This endpoint supports sending reference data in requests. To learn how you can retrieve this data when polling, refer to [Returning reference data for asynchronous requests](#).

Sample Request Body:

```
{
  "mergeTriggerRecordData": {
    "fieldNames": [
      "EMAIL_ADDRESS_"
    ],
    "mergeTriggerRecords": [
      {
        "fieldValues": [
          "john.doe@oracle.com"
        ],
        "optionalData": [
          {
            "name": "CITY_",
            "value": "Redwood City"
          },
          {
            "name": "MOBILE_NUMBER_",
            "value": "12345678933"
          }
        ]
      }
    ],
    "fieldValues": [
      "george.washington@oracle.com"
    ]
  }
}
```

```

    ],
    "optionalData": [
      {
        "name": "CITY_",
        "value": "Vancouver"
      },
      {
        "name": "MOBILE_NUMBER_",
        "value": "12345678934"
      }
    ]
  }
],
},
"referenceData": {
  "key": "value"
},
"mergeRule": {
  "insertOnNoMatch": false,
  "updateOnMatch": "NO_UPDATE",
  "matchColumnName1": "EMAIL_ADDRESS_",
  "matchColumnName2": "",
  "matchColumnName3": "",
  "matchOperator": "NONE",
  "optinValue": "I",
  "optoutValue": "O",
  "htmlValue": "H",
  "textValue": "T",
  "rejectRecordIfChannelEmpty": "E",
  "defaultPermissionStatus": "OPTIN"
}
}

```

Response Notes:

- The system validates the request payload syntax to ensure that the payload is well-formed JSON. However, it does not validate correctness of the values. For example, it does not validate if the campaign object exists or not, nor does it validate whether the payload uses the correct property names. Subsequently, you must poll for results to learn whether the

request succeeded or failed due to other reasons.

- A successful response for the asynchronous API returns an HTTPS status code of 202 Accepted.
- The response body contains the information you need to analyze the response that you would get from the Polling API.
 - requestId: Unique identifier of the API request. The value is alphanumeric.
 - apiName: Name of the API request that you triggered. Always asyncMergeTriggerEmail for this API request's response.
 - status: When successful, the status value is REQUEST_ACCEPTED. This means that the payload syntax was well-formed JSON and that the request was placed in the system queue for processing. You must poll for the final result; for more information, see [Polling API endpoint](#).
- See [Common error responses for the Asynchronous API](#) for details about the error responses.

Sample Successful Response:

```
{
  "requestId": "SkxIMG1YOjoxNTlxNzg3ODM0",
  "apiName": "asyncMergeTriggerEmail",
  "status": "REQUEST_ACCEPTED"

  "links": [
    {
      "rel": "self",
      "href": "/rest/asyncapi/v1.3/campaigns/myEmailCmpn/email",
      "method": "POST"
    },
    {
      "rel": "asyncResponse",
      "href": "/rest/asyncapi/v1.3/requests/SkxIMG1YOjoxNTlxNzg3ODM0",
      "method": "GET"
    }
  ]
}
```

Sample Error Response - Queue is Full:

```
{
  "type": "",
  "title": "Unable to submit Async request",
  "errorCode": "UNABLE_TO_SUBMIT_ASYNC_REQUEST",
  "detail": "Error in submitting aysnc request",
  "errorDetails": []
}
```

Asynchronous Merge Trigger SMS API

The Asynchronous Merge Trigger SMS API has a different service URL than the standard Merge Trigger SMS API, but it uses the same parameters, header, and request body. The response is different: it provides the data necessary to use the [Polling API endpoint](#) results.

Note: The Asynchronous API is generally available for all Responsys customers.

Service URL:

/rest/asyncApi/v1.3/campaigns/{campaignName}/sms

Required Path Parameters:

campaignName - Name of the SMS campaign to send to the recipients

Request Method:

POST

Request Header:

Authorization=<AUTH_TOKEN>

Content-Type=application/json

Message-Expiration=<ttl value – if omitted, the system uses the default of 3600 seconds. Use this value to set an expiration time for your request.>

Request Body Notes:

- You can learn about the required parameters in the [Merge Trigger Email REST Endpoints page](#). View the Request section of the topic.
- This endpoint supports sending reference data in requests. To learn how you can retrieve this data when polling, refer to [Returning reference data for asynchronous requests](#).

Sample Request Body:

```
{
  "mergeTriggerRecordData": {
    "fieldNames": [
      "CUSTOMER_ID_",
      "EMAIL_ADDRESS_",
      "MOBILE_NUMBER_",
      "MOBILE_COUNTRY_"
    ],
    "mergeTriggerRecords": [
      {
        "fieldValues": [
          "20",
          "john.doe@oracle.com",
          "16503930157",
          "US"
        ],
        "optionalData": [
          {
```

```

        "name": "CUSTOM_TEXT",
        "value": "Redwood City"
    },
    {
        "name": "CUSTOM_TEXT1",
        "value": " u00ea"
    }
]
},
{
    "fieldValues": [
        "21",
        "george.washington@oracle.com",
        "16502229405",
        "US"
    ],
    "optionalData": [
        {
            "name": "CUSTOM_TEXT",
            "value": "Blue Jeans"
        },
        {
            "name": "CUSTOM_TEXT1",
            "value": " u00ea"
        }
    ]
}
]
},
"referenceData": {
    "key": "value"
},
"mergeRule": {
    "insertOnNoMatch": true,
    "updateOnMatch": "REPLACE_ALL",
    "matchColumnName1": "EMAIL_ADDRESS_",
    "matchColumnName2": "",
    "matchColumnName3": "",
    "matchOperator": "NONE",
    "optinValue": "I",

```

```

"optoutValue": "O",
"htmlValue": "H",
"textValue": "T",
"rejectRecordIfChannelEmpty": "E",
"defaultPermissionStatus": "OPTIN"
}
}

```

Response Notes:

- The system validates the request payload syntax to ensure that the payload is well-formed JSON. However, it does not validate correctness of the values. For example, it does not validate if the campaign object exists or not, nor does it validate whether the payload uses the correct property names. Subsequently, you must poll for results to learn whether the request succeeded or failed due to other reasons.
- A successful response for the asynchronous API returns an HTTPS status code of 202 Accepted.
- The response body contains the information you need to analyze the response that you would get from the Polling API.
 - requestId: Unique identifier of the API request. The value is alphanumeric.
 - apiName: Name of the API request that you triggered. Always asyncMergeTriggerEmail for this API request's response.
 - status: When successful, the status value is REQUEST_ACCEPTED. This means that the payload syntax was well-formed JSON and that the request was placed in the system queue for processing. You must poll for the final result; for more information, see [Polling API endpoint](#).
- See [Common error responses for the Asynchronous API](#) for details about the error responses.

Sample Successful Response:

```

{
  "requestId": "SkxIMG1YOjoxNTIxNzg3ODM1",

```

```

    "apiName": "asyncMergeTriggerSms",
    "status": "REQUEST_ACCEPTED"

    "links": [
      {
        "rel": "self",
        "href": "/rest/asyncapi/v1.3/campaigns/mySMScmpn/sms",
        "method": "POST"
      },
      {
        "rel": "asyncResponse",
        "href": "/rest/asyncapi/v1.3/requests/SkxIMG1YOjoxNTlxNzg3ODM1",
        "method": "GET"
      }
    ]
  }
}

```

Sample Error Response - Queue is Full:

```

{
  "type": "",
  "title": "Unable to submit Async request",
  "errorCode": "UNABLE_TO_SUBMIT_ASYNC_REQUEST",
  "detail": "Error in submitting aysnc request",
  "errorDetails": []
}

```

Asynchronous Trigger a Custom Event

Use the following API to asynchronously 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).

Note: The Asynchronous API is generally available for all Responsys customers.

Service URL:

/rest/asyncApi/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:

```
{
  "requestId": "MGkybHBZOjoxNjlwMTI3ODAA0",
  "apiName": "asyncTriggerCustomEvent",
  "status": "REQUEST_ACCEPTED",
  "links": [
    {
```

```
"rel": "self",
"href": "/rest/asyncApi/v1.3/events/webservices",
"method": "POST"
},
{
"rel": "asyncResponse",
"href": "/rest/asyncApi/v1.3/requests/MGkybHBZOjoxNjlwMTI3ODAA0",
"method": "GET"
}
]
}
```

Asynchronous Trigger REI Event

Triggers a REI Event for Profile List, App Channel List, and Web Push recipients.

Enables developers to trigger REI events for users across email, print, and mobile app channels to begin Program Orchestration.

REI Events is a feature that needs to be enabled for your account. To access this feature for existing accounts, please log in to My Oracle Support and create a service request.

This endpoint is the asynchronous alternative to [Trigger REI Event](#).

Note: The Asynchronous API is generally available for all Responsys customers.

Service URL:

`/rest/asyncApi/v1.3/events/rei/{eventName}`

Required Path Parameters:

`eventName` - Name of the REI event to trigger.

Request Method:

POST

Request Header:

Authorization=<AUTH_TOKEN>

Content-Type=application/json

Request Body - Required Properties:

For more information about forming a request, see the [Trigger REI Event REST Endpoints](#) page in the *Oracle Responsys REST API Guide*. View the Request section of the topic.

Sample Request Body:

```
{
  "reiEvent": {
    "eventNumberDataMapping": null,
    "eventDateDataMapping": null,
    "eventStringDataMapping": null
  },
  "recipientData": [
    {
      "recipient": {
        "customerId": null,
        "emailAddress": "api.user@oracle.com",
        "emailSHA256Hash": null,
        "emailMD5Hash": null,
        "listName": {
          "folderName": "Events",
```

```

    "objectName": "Summer_Event_Attendees"
  },
  "recipientId": "",
  "mobileNumber": null,
  "emailFormat": "HTML_FORMAT"
},
"optionalData": [
  {
    "name": "DATEFIELD",
    "value": "2018-12-13T15:47:00.000+05:30"
  },
  {
    "name": "INTFIELD",
    "value": 1
  },
  {
    "name": "NUMFIELD",
    "value": 45
  },
  {
    "name": "TEXTFIELD",
    "value": "Attendee"
  }
]
}
]
}

```

Response Notes:

- A successful response for the asynchronous API returns an HTTPS status code of 202 Accepted.
- A successful response returns a requestId, to be used in the [Polling API endpoint](#) to retrieve the status of your request.
- See [Common error responses for the Asynchronous API](#) for details about the error responses.

Sample Response:

```
{
  "requestId": "Z1VzZUxrOjoxNjQzNjA4NTc0",
  "apiName": "asyncTriggerReiEvent",
  "status": "REQUEST_ACCEPTED",
  "links": [
    {
      "rel": "self",
      "href": "/rest/asyncApi/v1.3/events/rei/Summer_Event",
      "method": "POST"
    },
    {
      "rel": "asyncResponse",
      "href": "/rest/asyncApi/v1.3/requests/Z1VzZUxrOjoxNjQzNjA4NTc0",
      "method": "GET"
    }
  ]
}
```

Sample Error Response - REI event not found

```
{
  "response": {
    "errorCode": "REI_EVENT_NOT_FOUND",
    "detail": "Unable to retrieve EventDefinition for eventName:Summer_Event for account:70558:apitesting",
    "type": "",
    "title": "Rei event not found",
    "errorDetails": "[]"
  },
  "requestId": "TkoxdWtBOjoxNjQzNjA4MTEx",
  "apiName": "asyncTriggerReiEvent",
  "timeZone": "India Standard Time",
  "requestTime": "2022-01-31T11:18:31.352",
  "requestProcessedTime": "2022-01-31T11:18:31.630",
  "status": "ERROR"
}
```

Asynchronous Trigger Push Message API

The Asynchronous Trigger Push Message API has a different service URL than the standard [Trigger Push Message API](#), but it uses the same parameters, header, and request body. The response is different: it provides the data necessary to use the [Polling API endpoint](#) results.

Note: The Asynchronous API is generally available for all Responsys customers.

Service URL:

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

Required Path Parameters:

`campaignName` - Name of the push campaign to send to the recipients

Request Method:

POST

Request Header:

Authorization=<AUTH_TOKEN>

Content-Type=application/json

Message-Expiration=<ttl value – if omitted, the system uses the default of 3600 seconds. Use this value to set an expiration time for your request.>

Request Body - Required Properties:

See the [Trigger Push Message Endpoints](#) page in the Oracle Responsys REST API Guide. View the Request body section of the topic.

Request Notes:

- You can send push messages to existing Responsys Push campaigns for to up to 200 members of a Profile List and its corresponding 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 `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": "PROFILE"` when recipients need to be matched from Profile List (that is, known recipients).
 - 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": "PUSH"` when recipients need to be matched from App Channel List (that is, unknown recipients).
 - 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": "PUSH"`, the App Channel List attributes that you must specify are both `deviceId` and `apiKey`. Including both `deviceId` and `apiKey` 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 is used for customizing 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 currency symbol is escaped as `\u20AC`, the yen currency symbol is escaped as `\u00A5`, an umlauted u is escaped as `\u00FC`, an accented e is escaped as `\u00E9`, and the like. Otherwise, you may receive an `INVALID_REQUEST_CONTENT` error.
- This endpoint supports sending reference data in requests. To learn how you can retrieve this data when polling, refer to [Returning reference data for asynchronous requests](#).

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` value, 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 and app combinations that the known mobile app user has installed.
- In your code, ensure that `deviceId` and `apiKey` contain the actual values.

```
{
  "recipientData": [
    {
      "customerId": null,
      "emailAddress": null,
      "recipientId": 131006807,
      "mobileNumber": null,
      "emailSHA256Hash": null,
      "emailMD5Hash": null,
      "deviceId": "",
      "apiKey": "",
      "listType": "PROFILE",
      "optionalData": [
        {
          "name": "Brand",
          "value": "Durham"
        },
        {
          "name": "Fit",
          "value": "Tailored"
        }
      ]
    }
  ]
}
```

```

    }
  ]
},
{
  "customerId": null,
  "emailAddress": null,
  "recipientId": 100007327,
  "mobileNumber": null,
  "emailSHA256Hash": null,
  "emailMD5Hash": null,
  "deviceId": "",
  "apiKey": "",
  "listType": "PROFILE",
  "optionalData": [
    {
      "name": "Cost",
      "value": "100.00"
    },
    {
      "name": "CUSTOM2",
      "value": "custom_value"
    }
  ]
}
],
"referenceData": {
  "data": [
    {
      "key1": "value"
    },
    {
      "key2": "value"
    }
  ]
}
}

```

Response Notes:

- The system validates the request payload syntax to ensure that the payload is well-formed JSON. However, it does not validate correctness of the values. For example, it does not validate if the campaign object exists or not, nor does it validate whether the payload uses the correct property names. Subsequently, you must poll for results to learn whether the request succeeded or failed due to other reasons.
- A successful response for the asynchronous API returns an HTTPS status code of 202 Accepted.
- The response body contains the information you need to analyze the response that you would get from the Polling API.
 - requestId: Unique identifier of the API request. The value is alphanumeric.
 - apiName: Name of the API request that you triggered. Always asyncMergeTriggerEmail for this API request's response.
 - status: When successful, the status value is REQUEST_ACCEPTED. This means that the payload syntax was well-formed JSON and that the request was placed in the system queue for processing. You must poll for the final result; for more information, see [Polling API endpoint](#).
- See [Common error responses for the Asynchronous API](#) for details about the error responses.

Sample Successful Response:

```
{
  "requestId": "dXRqYUhDOjoxNjE3Nzg1OTM4",
  "apiName": "asyncTriggerPush",
  "status": "REQUEST_ACCEPTED",
  "links": [
    {
      "rel": "self",
      "href": "/rest/asyncApi/v1.3/campaigns/Push_Campaign/push",
      "method": "POST"
    },
    {
```

```
"rel": "asyncResponse",
"href": "/rest/asyncApi/v1.3/requests/dXRqYUhDOjoxNjE3Nzg1OTM4",
"method": "GET"
}
]
}
```

Sample Error Response - Queue is Full:

```
{
  "type": "",
  "title": "Unable to submit Async request",
  "errorCode": "UNABLE_TO_SUBMIT_ASYNC_REQUEST",
  "detail": "Error in submitting aysnc request",
  "errorDetails": []
}
```

Polling API endpoint

Use the Polling API to find out the status of up to 10 asynchronous requests.

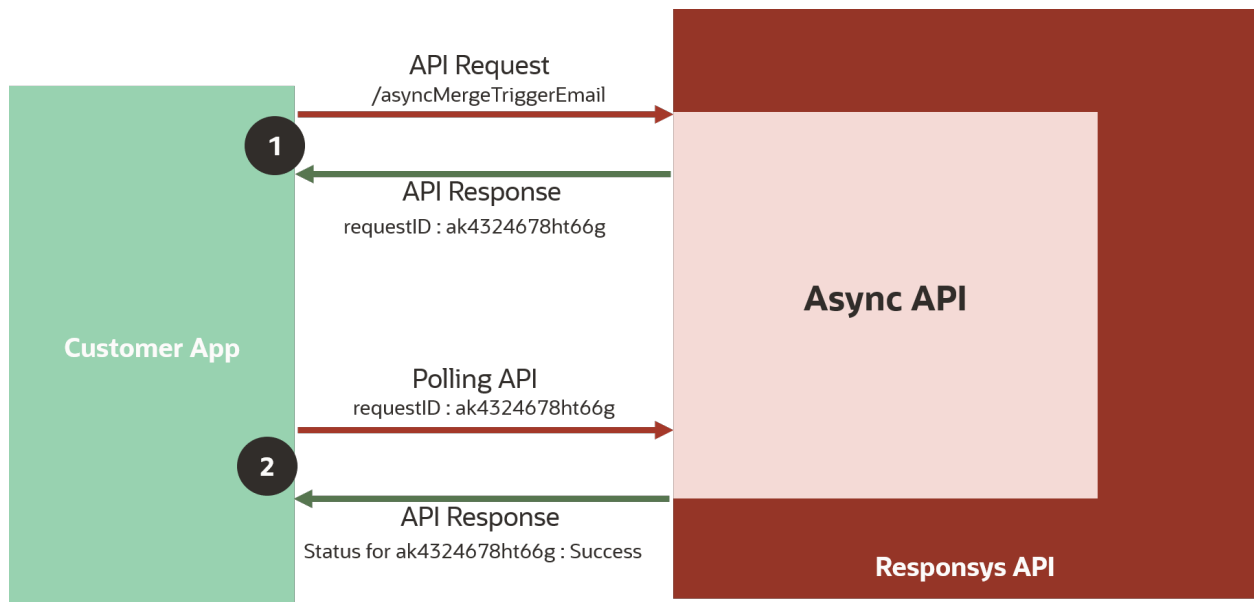
 **Note:** The Asynchronous API is generally available for all Responsys customers.

Workflow

Leveraging the asynchronous APIs and the polling API typically resembles this workflow:

1. A client sends a request to a Responsys asynchronous API endpoint. The asynchronous API endpoint responds to the client with a request ID.
2. A client sends the request ID to the Polling API endpoint to find out the status of the request. The Polling API responds with the status of the asynchronous request.

Here's a diagram illustrating how the polling API works:



Optionally you can include reference data in the request to the asynchronous endpoint. The Polling API can then return the reference data specified in the asynchronous request. Refer to [Returning reference data for asynchronous requests](#) for more information.

Request details

- You can specify up to 10 request IDs per request.
- Client applications must poll for responses within 24 hours after the initial API request was processed. A daily job on the server deletes responses that are more than one day old.

- The system throttles the Polling API. You can get the throttling limit of the API by using the [Get Throttling Limits](#) REST API request.
- Status returned by the polling API is only about completion of the merge and submitting to the personalization queue. The results do not reflect the final campaign send status. In other words, it's possible that the campaign send could fail due to personalization errors even if the result indicates success.

Request Method	Service URL
GET	/rest/asyncApi/v1.3/requests/{requestId}
Request Headers	Authorization=<AUTH_TOKEN> Content-Type=application/json
Path Parameters	• requestId: Unique identifier of the API request. Obtain this value from the successful response to your asynchronous request. You can specify up to 10 request IDs per request, combined using the "&" operator.
Query Parameters	• ?returnReferenceData=true: If specified, the response includes a referenceData property. The value of referenceData mirrors the reference data specified in the original asynchronous request. Refer to Returning reference data for asynchronous requests for more information. Returning reference data is supported by the following endpoints: • Asynchronous Merge Trigger Email API • Asynchronous Merge Trigger SMS API • Asynchronous Trigger Push Message API
Request Body Required Properties	N/A

Example Requests

Retrieve the polling results for the request IDs `dzBPSUZGOjoxNTIxNDQ0NjUz` and `dzBBGFFDcvuTNDTYNVDr`.

```
/rest/asyncApi/v1.3/requests/dzBPSUZGOjoxNTIxNDQ0NjUz&dzBBGFFDcvuTNDTYNVDr
```

Retrieve the polling results for the request ID `dzBPSUZGOjoxNTIxNDQ0NjUz` and display the reference data for the asynchronous request.

```
/rest/asyncApi/v1.3/requests/dzBPSUZGOjoxNTIxNDQ0NjUz?returnReferenceData=true
```

Response Notes:

The response HTTP STATUS code is always 200 (Ok), unless there is a system error (500 Internal Server Error). See [Common error responses for the Polling API](#) for details about the error responses.

For a successful response, the response body properties are as follows:

- `responseMessage`: object that contains either the merge results for the recipients sent in the async Merge Trigger request, a message property with the in-progress message "PROCESSING_REQUEST", or the standard REST API error JSON. See the [Handling errors](#) section for more details about error responses.
- `requestId`: the unique identifier that was returned by the original asynchronous API request.
- `apiName`: Property that contains the name of the request API.
- `timeZone`: The time zone of the Responsys account performing the request.
- `requestTime`: Timestamp of when the request was submitted.
- `requestProcessedTime`: Timestamp when the request was processed.
- `status`: Status of the async API request.

Sample Responses - Successful

Merge successful. In this scenario, two Async Merge Trigger Email requests were successfully processed, and a response code of 200 OK was received.

```
"requestData": [
  {
    "responseMessage": [
      {
        "recipientId": 16005207,
        "success": true,
        "errorMessage": ""
      },
      {
        "recipientId": 21001007,
        "success": true,
        "errorMessage": ""
      }
    ],
    "requestId": "dzBPSUZGOjoxNTIxNDQ0NjUz",
    "apiName": "asyncMergeTriggerEmail",
    "timeZone": "Brasilia Time",
    "requestTime": "2021-01-13 06:36:25.774",
    "requestProcessedTime": "2021-01-13 06:36:27",
    "status": "SUCCESS"
  },
  {
    "responseMessage": [
      {
        "recipientId": 81006008,
        "success": true,
        "errorMessage": ""
      },
      {
        "recipientId": 81006009,
        "success": true,
        "errorMessage": ""
      }
    ],
  },
]
```

```

    "requestId": "dzBBGFFDcvuTNDTYNVDr",
    "apiName": "asyncMergeTriggerEmail",
    "timeZone": "Brasilia Time",
    "requestTime": "2021-01-13 06:16:03.221",
    "requestProcessedTime": "2021-01-13 06:26:01",
    "status": "SUCCESS"
  }
]

```

Merge successful with returnReferenceData enabled. This scenario is the same as above, with displaying the reference data of the original request.

```

{
  "response": [
    {
      "success": true,
      "errorMessage": null,
      "recipientId": 7001327
    },
    {
      "success": false,
      "errorMessage": "FAILURE: Record 1 = Bad Email Format\r\n",
      "recipientId": 0
    }
  ],
  "referenceData": "{ \"referenceKey1\": \"referenceValue1\" }",
  "requestId": "ZXFBNWRGOjoxNjlyNDY4NzQx",
  "apiName": "asyncMergeTriggerSms",
  "timeZone": "India Standard Time",
  "requestTime": "2021-05-31T19:15:41.957",
  "requestProcessedTime": "2021-05-31T19:15:48.363",
  "status": "SUCCESS"
}

```

Mixed results for the merge. In this scenario, the request succeeded for the first recipient, but failed for the second recipient.

```
{
  "responseMessage": [
    {
      "recipientId": 16005207,
      "success": true,
      "errorMessage": ""
    },
    {
      "recipientId": 0,
      "success": false,
      "errorMessage": "FAILURE: Record 1 = INVALID_PARAMETER:matchColumnName1 is
not found in fieldNames or corresponding value is null or empty\r\n"
    }
  ],
  "requestId": "MWxNQXJDOjoxNTlwOTQ4ODI1",
  "apiName": "asyncMergeTriggerSms",
  "timeZone": "Brasilia Time",
  "requestTime": "2018-03-13 06:47:05.435",
  "requestProcessedTime": "2018-03-13 06:47:06",
  "status": "SUCCESS"
}
```

Processing request. In this scenario, an async request is still being processed, and a response code of 200 OK was received.

```
{
  "responseMessage": {
    "message": "PROCESSING_REQUEST"
  },
  "requestId": "SkxIMG1YOjoxNTlxNzg3ODM1",
  "apiName": "",
  "timeZone": "Brasilia Time",
  "requestTime": "2018-03-14 04:49:57.0",
  "requestProcessedTime": "",
  "status": "IN_PROGRESS"
}
```

Sample Response - Failure:

The following example shows a sample error response – in this case, the `responseMessage` object contains the error JSON, which tells us that the campaign was invalid. A response code of 200 OK is shown, however, because the polling request itself completed successfully.

```
{
  "responseMessage": {
    "errorCode": "CAMPAIGN_IS_INVALID",
    "detail": "An error has occurred and the system is unable to process your request at this time. Please contact Responsys Support for further assistance.",
    "type": "",
    "title": "Not a valid campaign",
    "errorDetails": "[]"
  },
  "requestId": "dzBPSUZGOjoxNTIxNDQ0NjUz",
  "apiName": "asyncMergeTriggerEmail",
  "timeZone": "Brasilia Time",
  "requestTime": "2018-03-19 13:00:53.013",
  "requestProcessedTime": "2018-03-19 13:00:55",
  "status": "ERROR"
}
```

Multiple Polling API endpoint

Use the Multiple Polling API to find out the status of up to 200 asynchronous requests.

Note: The Asynchronous API is generally available for all Responsys customers.

Request details

- You can specify up to 200 request IDs per request.
- Client applications must poll for responses within 24 hours after the initial API request was processed. A daily job on the server deletes responses that are more than one day old.
- The system throttles the Polling API. You can get the throttling limit of the API by using the [Get Throttling Limits](#) REST API request.
- Status returned by the polling API is only about completion of the merge and submitting to the personalization queue. The results do not reflect the final campaign send status. In other words, it's possible that the campaign send could fail due to personalization errors even if the result indicates success.

Request Method	Service URL
POST	/rest/asyncApi/v1.3/requests
Request Headers	Authorization=<AUTH_TOKEN> Content-Type=application/json
Request Body Required Properties	• <code>requestIds</code> - Array of request IDs from sending asynchronous requests. Specify up to 200 request IDs per request.

Example Request

Retrieve the polling results for the request IDs `QmU4dXpY0joxNTc5NzA0MDEw`, `OUJDVFhK0joxNTc5MTAyODg5`, and `OUJDVFhK0joxNTc5eeAyODg5`.

```
POST /rest/asyncApi/v1.3/requests
```

```
{
  "requestIds": [
    "QmU4dXpY0joxNTc5NzA0MDEw",
```

```
"OUJDVFhKOjoxNTc5MTAyODg5",  
"OUJDVFhKOjoxNTc5eeAyODg5"  
]  
}
```

Response Notes

The response HTTP STATUS code is always 200 (Ok), unless there is a system error (500 Internal Server Error). See [Common error responses for the Polling API](#) for details about the error responses.

For a successful response, the response body properties are as follows:

- response: object that contains either the merge results for the recipients, or the error response messages. See the [Common error responses for the Polling API](#) section for more details about error responses.
- requestId: the unique identifier that was returned by the original asynchronous API request.
- apiName: Property that contains the name of the request API.
- timeZone: The time zone of the Responsys account performing the request.
- requestTime: Timestamp of when the request was submitted.
- requestProcessedTime: Timestamp when the request was processed.
- status: Status of the async API request. See the [Common error responses for the Polling API](#) section for more details about error responses.

Sample Responses - Successful

Mixed results for the merge. In this scenario, the request succeeded for the first recipient, but failed for the second, and third recipients.

```
[
  {
    "response": {
      "recordData": {
        "records": [
          [
            "122309767"
          ]
        ],
        "mapTemplateName": null,
        "fieldNames": [
          "RIID_"
        ]
      },
      "mergeRule": {
        "optoutValue": "O",
        "rejectRecordIfChannelEmpty": "E",
        "updateOnMatch": "REPLACE_ALL",
        "optinValue": "I",
        "textValue": "T",
        "htmlValue": "H",
        "matchColumnName2": "",
        "insertOnNoMatch": true,
        "defaultPermissionStatus": "OPTIN",
        "matchOperator": "AND",
        "matchColumnName1": "CUSTOMER_ID_"
      }
    },
    "requestId": "QmU4dXpYOjoxNTc5NzA0MDEw",
    "apiName": "asyncMergeListMembers",
    "timeZone": null,
    "requestTime": "2020-01-22 06:40:10.779",
    "requestProcessedTime": "2020-01-22 06:40:10.822",
    "status": "SUCCESS"
  },
  {
    "response": {
      "type": "",
      "title": "Response purged",
      "errorCode": "RESPONSE_PURGED",
    }
  }
]
```

```

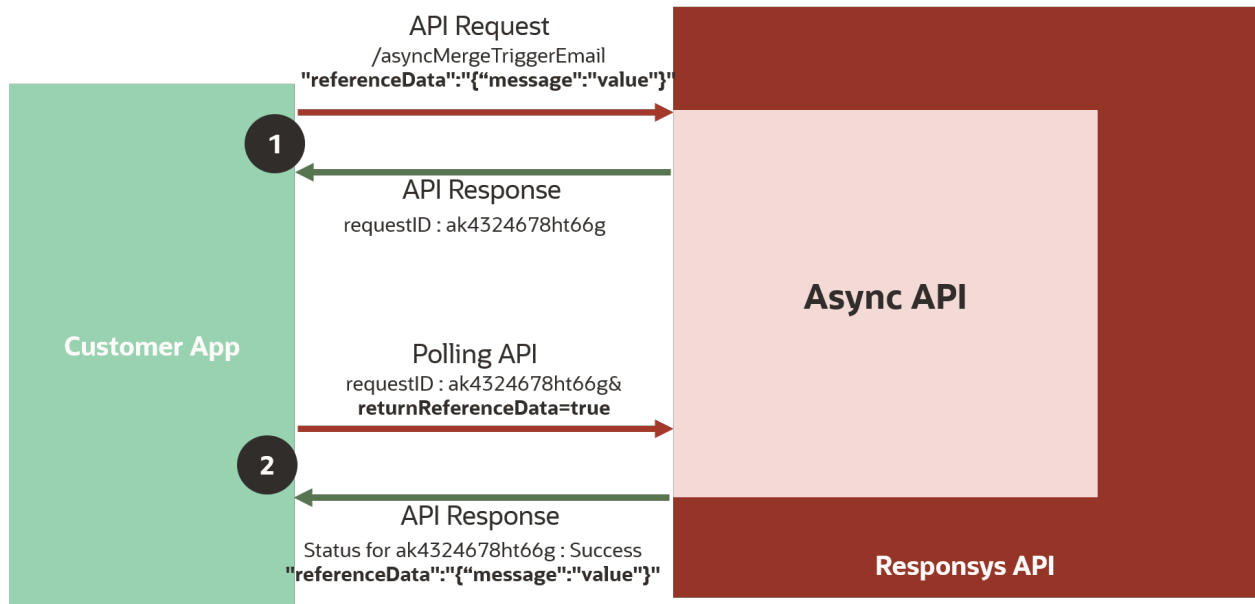
    "detail": "Response Message Purged",
    "errorDetails": "[]"
  },
  "requestId": "OUJDVFhKOjoxNTc5MTAyODg5",
  "apiName": "",
  "timeZone": null,
  "requestTime": "2020-01-15 07:41:29.0",
  "requestProcessedTime": "",
  "status": "UNKNOWN"
},
{
  "response": {
    "type": "",
    "title": "Invalid Request Id",
    "errorCode": "INVALID_REQUEST_ID",
    "detail": "Request Id is not valid ",
    "errorDetails": "[]"
  },
  "requestId": "OUJDVFhKOjoxNTc5eeAyODg5",
  "apiName": null,
  "timeZone": null,
  "requestTime": null,
  "requestProcessedTime": null,
  "status": "ERROR"
}
]

```

Returning reference data for asynchronous requests

Reference data is additional data you can include when making asynchronous API requests. After calling an asynchronous API endpoint with your reference data, the polling API can return that data sent in the original asynchronous request.

Here's a diagram that illustrates how this works.



Workflow

1. Send an asynchronous request to the merge trigger email API [endpoint](#), where my campaign name is "Summer_Campaign".

```
POST /rest/asyncApi/v1.3/campaigns/Summer_Campaign/email
```

Include the following reference data in the request:

```
"referenceData": "sample_ID : 173289739"
```

Request body

```
{
  "mergeTriggerRecordData": {
    "fieldNames": [
      "EMAIL_ADDRESS_"
```

```
],
"mergeTriggerRecords": [
  {
    "fieldValues": [
      "john.doe@oracle.com"
    ],
    "optionalData": [
      {
        "name": "CITY_",
        "value": "Redwood City"
      },
      {
        "name": "MOBILE_NUMBER_",
        "value": "12345678933"
      }
    ]
  },
  {
    "fieldValues": [
      "george.washington@oracle.com"
    ],
    "optionalData": [
      {
        "name": "CITY_",
        "value": "Vancouver"
      },
      {
        "name": "MOBILE_NUMBER_",
```

```
        "value": "12345678934"
      }
    ]
  }
]
},
"referenceData": "sample_ID : 173289739",
"mergeRule": {
  "insertOnNoMatch": false,
  "updateOnMatch": "NO_UPDATE",
  "matchColumnName1": "EMAIL_ADDRESS_",
  "matchColumnName2": "",
  "matchColumnName3": "",
  "matchOperator": "NONE",
  "optinValue": "I",
  "optoutValue": "O",
  "htmlValue": "H",
  "textValue": "T",
  "rejectRecordIfChannelEmpty": "E",
  "defaultPermissionStatus": "OPTIN"
}
}
```

Response

If successful, a request ID is returned. In step 2 we will send this request ID to the Polling API to find out the status of the asynchronous request.

```
{
  "requestId": "SkxIMG1YOjoxNTIaNzg3ODM0",
  "apiName": "asyncMergeTriggerEmail",
  "status": "REQUEST_ACCEPTED" "links": [
    {
      "rel": "self",
      "href": "/rest/asyncapi/v1.3/campaigns/myEmailCmpn/email",
      "method": "POST"
    },
    {
      "rel": "asyncResponse",
      "href": "/rest/asyncapi/v1.3/requests/SkxIMG1YOjoxNTIaNzg3ODM0",
      "method": "GET"
    }
  ]
}
```

2. Send a request to the [Polling API](#) to find out the status of the asynchronous request.

Request

```
GET
/rest/asyncApi/v1.3/requests/SkxIMG1YOjoxNTIaNzg3ODM0?returnReferenceData
=true
```

Response

The response includes information about the asynchronous request and the reference data specified. See [Polling API endpoint](#) for more information.

```
{
  "response": [
    {
      "success": true,
      "errorMessage": null,
      "recipientId": 7001327
    }
  ],
  "referenceData": "sample_ID:173289739",
  "requestId": "SkxIMG1YOjoxNTIxBzg3ODM0",
  "apiName": "asyncMergeTriggerSms",
  "timeZone": "India Standard Time",
  "requestTime": "2021-05-31T19:15:41.957",
  "requestProcessedTime": "2021-05-31T19:15:48.363",
  "status": "SUCCESS"
}
```

Returning reference data is supported by the endpoints:

- [Asynchronous Merge Trigger Email API](#)
- [Asynchronous Merge Trigger SMS API](#)
- [Asynchronous Trigger Push Message API](#)

Asynchronous Reporting API

Note: The Asynchronous API is generally available for all Responsys customers.

An asynchronous API for aggregated reporting data that enables you to pull campaign performance metrics for all the Responsys channels (Email, SMS, and Mobile Apps). This API is released under our Controlled Availability program. You can request access to this feature by submitting a request to [My Oracle Support](#). Some of the benefits of leveraging this API include:

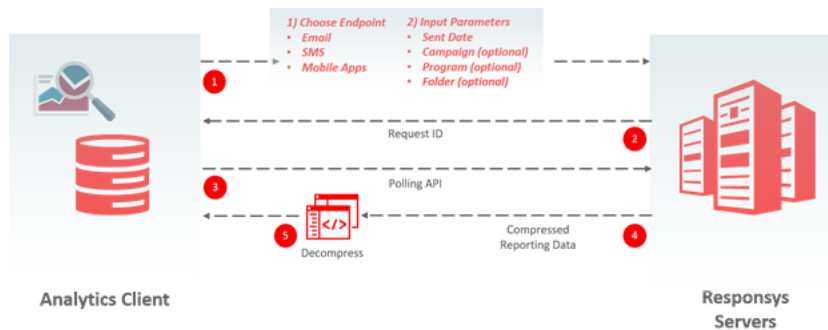
- Build custom visuals to display Responsys data
- Integrate Responsys data with other business applications
- Automate complex reporting tasks to save time
- Data matches with Responsys Reports (Interactive Dashboards)

Important considerations:

- The date range filter cannot exceed a 30 day span. An error will be returned if the filter date range exceeds 30 days. Data can however be retrieved from up to 2 calendar years from the current date
- Requests will timeout after 5 minutes
- You can submit up to 5 requests per minute
- Row output is limited to 10,000 rows

Workflow

The following diagram illustrates the workflow for the Reporting API.



Create a request for Email campaign performance metrics

Note: The Asynchronous API is generally available for all Responsys customers.

Service URL:

/rest/asyncApi/v1.3/performanceAnalytics/campaigns/email

Request Method:

POST

Request Header:

Authorization=<AUTH_TOKEN>

Content-Type=application/json

Request Body - Required Properties:

`sendFromDate` – Start date from which data is required. In the format YYYY-MM-DD.

`sendToDate` – End date to which data is required. In the format YYYY-MM-DD.

Request Body - Optional Properties:

`campaignName` – Name of the campaign.

`programName` – Name of the program.

`folderName` – Name of the folder where the campaign is located.

Sample Request Body:

```
{
  "sendFromDate":"2018-09-10",
  "sendToDate":"2018-09-12",
  "campaignName":"Test Campaign",
  "programName":"Test Program",
  "folderName":"Test Folder"
}
```

Response Notes:

- A successful response for the asynchronous API returns an HTTPS status code of 202 Accepted.
- A successful response returns a requestId, to be used in the [Polling API endpoint](#) to retrieve the status of your request.
- See [Common error responses for the Asynchronous API](#) for details about the error responses.

Sample Response - Successful

```
{
  "requestId": "UHplRHFuOjoxNTUxNzl4Mjc3",
  "apiName": "asyncPerformanceAnalyticsEmail",
  "status": "REQUEST_ACCEPTED",
  "links": [
    {
      "rel": "self",
      "href": "/rest/asyncApi/v1.3/performanceAnalytics/campaigns/email",
      "method": "POST"
    },
    {
      "rel": "asyncResponse",
      "href":
"/rest/asyncApi/v1.3/performanceAnalytics/requests/UHplRHFuOjoxNTUxNzl4Mjc3",
      "method": "GET"
    }
  ]
}
```

Create a request for SMS campaign performance metrics

Note: The Asynchronous API is generally available for all Responsys customers.

Service URL:

/rest/asyncApi/v1.3/performanceAnalytics/campaigns/sms

Request Method:

POST

Request Header:

Authorization=<AUTH_TOKEN>

Content-Type=application/json

Request Body - Required Properties:

`sendFromDate` – Start date from which data is required. In the format YYYY-MM-DD.

`sendToDate` – End date to which data is required. In the format YYYY-MM-DD.

Request Body - Optional Properties:

`campaignName` – Name of the campaign.

`programName` – Name of the program.

`folderName` – Name of the folder where the campaign is located.

Sample Request Body:

```
{
  "sendFromDate":"2018-09-10",
  "sendToDate":"2018-09-12",
  "campaignName":"Test Campaign",
  "programName":"Test Program",
  "folderName":"Test Folder"
}
```

Response Notes:

- A successful response for the asynchronous API returns an HTTPS status code of 202 Accepted.
- A successful response returns a requestId, to be used in the [Polling API endpoint](#) to retrieve the status of your request.

- See [Common error responses for the Asynchronous API](#) for details about the error responses.

Sample Response - Successful

```
{
  "requestId": "dWt2ZU9zOjoxNTUxNzI4OTA0",
  "apiName": "asyncPerformanceAnalyticsSms",
  "status": "REQUEST_ACCEPTED",
  "links": [
    {
      "rel": "self",
      "href": "/rest/asyncApi/v1.3/performanceAnalytics/campaigns/sms",
      "method": "POST"
    },
    {
      "rel": "asyncResponse",
      "href":
"/rest/asyncApi/v1.3/performanceAnalytics/requests/dWt2ZU9zOjoxNTUxNzI4OTA0",
      "method": "GET"
    }
  ]
}
```

Create a request for Mobile app campaign performance metrics

Note: The Asynchronous API is generally available for all Responsys customers.

Service URL:

/rest/asyncApi/v1.3/performanceAnalytics/campaigns/mobileapps

Request Method:

POST

Request Header:

Authorization=<AUTH_TOKEN>

Content-Type=application/json

Request Body - Required Properties:

`sendFromDate` – Start date from which data is required. In the format YYYY-MM-DD.

`sendToDate` – End date to which data is required. In the format YYYY-MM-DD.

Request Body - Optional Properties:

`campaignName` – Name of the campaign.

`programName` – Name of the program.

`folderName` – Name of the folder where the campaign is located.

Sample Request Body:

```
{
  "sendFromDate":"2018-09-10",
  "sendToDate":"2018-09-12",
  "campaignName":"Test Campaign",
  "programName":"Test Program",
  "folderName":"Test Folder"
}
```

Response Notes:

- A successful response for the asynchronous API returns an HTTPS status code of 202 Accepted.

- A successful response returns a requestId, to be used in the [Polling API endpoint](#) to retrieve the status of your request.
- See [Common error responses for the Asynchronous API](#) for details about the error responses.

Sample Response - Successful

```
{
  "requestId": "SmNWTGdaOjoxNTUxNzI5MjY1",
  "apiName": "asyncPerformanceAnalyticsMobileApps",
  "status": "REQUEST_ACCEPTED",
  "links": [
    {
      "rel": "self",
      "href": "/rest/asyncApi/v1.3/performanceAnalytics/campaigns/mobileapps",
      "method": "POST"
    },
    {
      "rel": "asyncResponse",
      "href":
"/rest/asyncApi/v1.3/performanceAnalytics/requests/SmNWTGdaOjoxNTUxNzI5MjY1",
      "method": "GET"
    }
  ]
}
```

Polling API endpoint

The Polling API is used to retrieve the reporting data from a request. The Polling API endpoint will respond to successful requests with compressed gzip. The response must be uncompressed to retrieve the data.

Note: The Asynchronous API is generally available for all Responsys customers.

Service URL:

`/rest/asyncApi/v1.3/performanceAnalytics/requests/{requestId}`

Required Path Parameters:

`requestId` - Unique identifier for the API request. Obtain this value from the successful response to your async request.

Request Method:

GET

Request Header:

Authorization=<AUTH_TOKEN>

Content-Type=application/json

Request Body - Required Properties:

Not applicable

Sample Request Body:

Not applicable

Response Notes:

- A successful response for the asynchronous API returns an HTTPS status code of 202 Accepted.
- See [Common error responses for the Polling API](#) for details about the error responses.
- Note that the date and time values consider the Responsys account's time zone
- The Polling API endpoint will respond to successful requests with compressed gzip. The response must be uncompressed to retrieve the data. There are various methods to uncompress the data.
 - If using API development tools - The response can be saved and downloaded as gzip. Unzip the file to view uncompressed data.
 - If integrating web services - The header "Responsys-Content-Encoding: gzip" indicates the response is encoded using gzip and needs to be decompressed.

```
private static String decompress(byte[] compressed) throws IOException {
    ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(compressed);
    GZIPInputStream gzipInputStream = new GZIPInputStream(byteArrayInputStream);
    BufferedReader bufferedReader = new BufferedReader(new InputStreamReader
(gzipInputStream, ENCODING));
    StringBuilder uncompressedData = new StringBuilder();
    String line;
    while ((line = bufferedReader.readLine()) != null) {
        uncompressedData.append(line);
    }
    bufferedReader.close();
    gzipInputStream.close();
    byteArrayInputStream.close();
    return uncompressedData.toString();
}
```

Sample Response - Successful

```
{
  "response": {
    "filterCriteria": {
      "sentDateFrom": "2018-10-01",
      "sentDateTo": "2018-10-05",
      "subjectArea": "Email"
    },
    "analyticsData": {
      "performanceMetrics": [
        {
          "conversions": 42,
          "unsubscribes": 4,
          "purpose": "PROMOTIONAL",
          "subjectLine": "Jump in NOW for this Deal",
          "launchCode": "qasmscamp.77461.81321",
          "delivered": 766,
          "uniqueClickToOpenRate": 16.54,
          "deliveredRate": 85.78,
          "responders": 19,
          "bounceRate": 14.22,
          "launchID": 81321,
          "sentDate": "2018-10-05",
          "revenue": 1123.79,
          "revenuePer1KEmail": 1467.09,
          "unsubscribeRate": 0.52,
          "uniqueClickThroughRate": 5.48,
          "marketingProgram": "New Products",
          "clickPerResponder": 2.32,
          "uniqueClicks": 40,
          "launchType": "PROGRAM TRIGGERED",
          "skipRate": 0,
          "openRate": 27.02,
          "averageClicks": 2,
          "externalCampaignID": "Not Applicable",
          "averageOrderValue": 26.76,
          "clickThroughRate": 10.97,
          "organizationUnit": "(Unspecified)",
          "bounced": 127,
```

```

"sent": 893,
"conversionRate": 5.48,
"failureRate": 0,
"uniqueOpens": 253,
"responderRate": 1.03,
"folder": "US West Campaigns",
"stage": "Send Email Campaign",
"uniqueOpenRate": 33.03,
"campaign": "Updated Recommendations 810",
"marketingStrategy": "Cross Sell",
"opens": 207,
"clicks": 84,
"uniqueConversions": 21,
"totalOrders": 1,
"uniqueConversionRate": 2.74,
"account": "responsysdemo"
},
{
"skipRate": 0,
"unsubscribes": 8,
"externalCampaignID": "Not Applicable",
"purpose": "PROMOTIONAL",
"subjectLine": "Jump in NOW for this Deal",
"organizationUnit": "SF North",
"launchCode": "qasmscamp.728101.535501",
"delivered": 5166,
"uniqueClickToOpenRate": 16.54,
"program": "Best Deal Alerts 15",
"sent": 5166,
"deliveredRate": 100,
"responders": 20,
"launchID": 535501,
"failureRate": 0,
"uniqueOpens": 130,
"sentDate": "2018-10-01",
"folder": "Shop and Save Extra",
"stage": "Send email campaign",
"unsubscribeRate": 0.15,
"campaign": "MVT_Test_Clearance_or_Best_Value",
"marketingStrategy": "Retention",

```

```
"clicks": 0,  
"marketingProgram": "Retail Stores",  
"clickPerResponder": 2.32,  
"responderRate": 1.03,  
"account": "responsysdemo",  
"launchType": "PROGRAM TRIGGERED"  
},  
{  
  "conversions": 92,  
  "unsubscribes": 4,  
  "purpose": "PROMOTIONAL",  
  "subjectLine": "Don't Miss This - Top Brands for Less",  
  "launchCode": "qasmscamp.490861.280141",  
  "delivered": 2975,  
  "uniqueClickToOpenRate": 16.54,  
  "program": "Welcome - Newsletter 19",  
  "deliveredRate": 91.59,  
  "responders": 18,  
  "bounceRate": 8.41,  
  "launchID": 280141,  
  "sentDate": "2018-10-03",  
  "revenue": 3255.42,  
  "revenuePer1KEmail": 1094.26,  
  "unsubscribeRate": 0.13,  
  "uniqueClickThroughRate": 3.06,  
  "marketingProgram": "Retail Stores",  
  "clickPerResponder": 10.47,  
  "launchType": "PROGRAM TRIGGERED",  
  "skipRate": 0,  
  "openRate": 29.61,  
  "averageClicks": 2,  
  "externalCampaignID": "Not Applicable",  
  "averageOrderValue": 35.39,  
  "clickThroughRate": 6.12,  
  "organizationUnit": "US",  
  "bounced": 273,  
  "sent": 3248,  
  "conversionRate": 3.09,  
  "failureRate": 0,  
  "uniqueOpens": 546,
```

```
"responderRate": 0.51,
"folder": "Trending Now",
"stage": "Send email campaign",
"uniqueOpenRate": 18.35,
"campaign": "Trending Now Newsletter GT-1_1",
"marketingStrategy": "Tier 2 Markets",
"opens": 881,
"clicks": 182,
"uniqueConversions": 46,
"totalOrders": 9,
"uniqueConversionRate": 1.55,
"account": "responsysdemo"
}
],
"timeZone": "Australian Eastern Standard Time (New South Wales)",
"subjectArea": "Email",
"rowCount": 582
},
"requestId": "dHNYUFhJOjoxNTQ0NDI5Mjk2",
"apiName": "asyncPerformanceAnalyticsEmail",
"requestTime": "2020-10-27T09:05:44.953",
"requestProcessedTime": "2020-10-27T09:05:49.359",
"status": "SUCCESS"
}
```

Handling errors

This section shows error messages specific to the Asynchronous REST API. Depending on the request, there may be other errors returned; for example, if the authorization token has expired, you may see the standard REST API error response `AUTHENTICATION_FAILED`. For more information about error types in the Responsys REST API, see the [Handling Errors](#) topic.

Common error responses for the Asynchronous API

These are common error responses for the Asynchronous API.

Request payload is not well-formed JSON (`INVALID_REQUEST_CONTENT`)

The system validates whether or not the request payload is well-formed JSON. If it is not, the system returns the following response with HTTPS Status Code 400 (Bad Request). Verify that your payload is in proper JSON format, and that it does not have missing or incorrectly used punctuation.

```
{
  "type": "",
  "title": "Invalid request content",
  "errorCode": "INVALID_REQUEST_CONTENT",
  "detail": "Request body is empty or invalid",
  "errorDetails": []
}
```

Recipient limit is exceeded (RECIPIENT_LIMIT_EXCEEDED)

If your client application submits an async request with more than 200 recipients, it is rejected immediately and is not put on the queue. Limit your requests to 200 or fewer recipients.

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

Queue is full (UNABLE_TO_SUBMIT_ASYNC_REQUEST)

If the request cannot be accepted due to queue being full, the following response is returned with HTTPS Status Code 500 (Internal Server Error). Try the request again later.

```
{
  "type": "",
  "title": "Unable to submit Async request",
  "errorCode": "UNABLE_TO_SUBMIT_ASYNC_REQUEST",
  "detail": "Error in submitting async request",
  "errorDetails": []
}
```

[Learn more](#)

Common error responses for the Polling API

Unless otherwise noted, each of the following has an HTTPS status code of 200 OK, because the polling request itself was considered successful, even if it contains an error response from the system.

Response purged (RESPONSE_PURGED)

The system returns this response when the async job status response is removed from the queue. Responses are retained in the system for 24 hours. Client applications must poll for responses within 24 hours after the initial API request was submitted.

```
{
  "response": {
    "type": "",
    "title": "Response purged",
    "errorCode": "RESPONSE_PURGED",
    "detail": "Response Message Purged",
    "errorDetails": "[]"
  },
  "requestId": "WGIJSWhsOjoxNTIxMzU5OTI1",
  "apiName": "",
  "timeZone": null,
  "requestTime": "2018-03-18 13:28:45.0",
  "requestProcessedTime": "",
  "status": "UNKNOWN"
}
```

Request expired while processing (ASYNC_REQUEST_EXPIRED)

The system returns this response when the async request has expired during processing. This happens if the time in the queue exceeds the Message-Expiration,

which is 3600 seconds by default.

```
{
  "responseMessage": {
    "errorCode": "ASYNC_REQUEST_EXPIRED",
    "detail": "Async request expired while processing.",
    "type": "",
    "title": "Async API request expired",
    "errorDetails": "[]"
  },
  "requestId": "WGIJSWhsOjoxNTlxMzU5OTI1",
  "apiName": "asyncMergeTriggerEmail",
  "requestTime": "2018-03-18 13:28:45.809",
  "requestProcessedTime": "2018-03-18 13:30:20",
  "status": "ERROR"
}
```

Request ID is not valid (INVALID_REQUEST_ID)

The system returns this response when the requestId provided is not valid. Verify the requestId is correct.

```
{
  "response": {
    "type": "",
    "title": "Invalid Request Id",
    "errorCode": "INVALID_REQUEST_ID",
    "detail": "Request Id is not valid ",
    "errorDetails": "[]"
  },
  "requestId": "VNTlwOTQ5MDcz",
  "apiName": null
  "requestTime": null,
  "requestProcessedTime": null,
  "status": "ERROR"
}
```

Common error responses for the Reporting API

Unless otherwise noted, each of the following has an HTTPS status code of 200 OK, because the polling request itself was considered successful, even if it contains an error response from the system.

Invalid request parameters (INVALID_PARAMETER)

The system returns this response when a request is missing one of the parameters `sendFromDate` and `sendToDate`, or both are missing.

```
{
  "type": "",
  "title": "Invalid request parameters",
  "errorCode": "INVALID_PARAMETER",
  "detail": "Send From Date and Send To Date are required",
  "errorDetails": []
}
```

The system returns this response when the `sendFromDate` is greater than the `sendToDate`.

```
{
  "type": "",
  "title": "Invalid request parameters",
  "errorCode": "INVALID_PARAMETER",
  "detail": "Send From Date can not be greater than Send To Date",
  "errorDetails": []
}
```

The system returns this response when the filter date range is more than 30 days.

```
{
  "type": "",
  "title": "Invalid request parameters",
  "errorCode": "INVALID_PARAMETER",
  "detail": "Send From Date and Send To Date difference should not be more than 30",
  "errorDetails": []
}
```

The system returns this response when the `sendFromDate` or `sendToDate` are not in the correct format `YYYY-MM-DD`.

```
{
  "type": "",
  "title": "Invalid date",
  "errorCode": "INVALID_DATE",
  "detail": "Date format should be yyyy-MM-dd",
  "errorDetails": []
}
```