API Technical Guide: Email Campaign Trigger

Cheetah Messaging

Table of Contents

1 Introduction

Purpose

The purpose of this document is to provide an overview of the EMAIL CAMPAIGN TRIGGER API endpoint within the Cheetah Messaging platform. This document discusses the intended use of the EMAIL CAMPAIGN TRIGGER endpoint, and provides technical details for how to implement the endpoint.

Overview

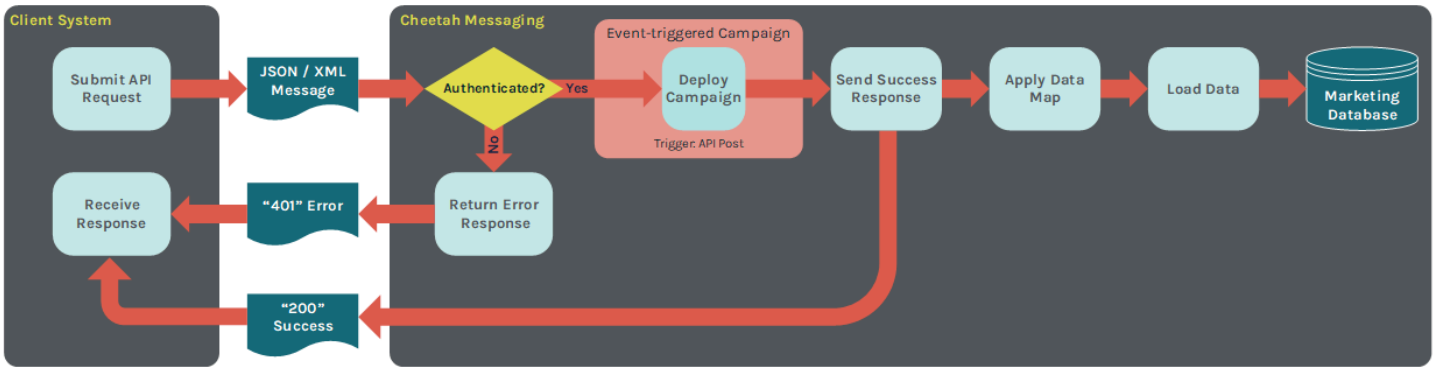
The EMAIL CAMPAIGN TRIGGER endpoint is used to trigger the deployment of an Event-triggered Campaign. This endpoint is sometimes referred to as the "instant trigger API" because the platform will build and send the email message immediately upon receipt of the API request, then load the data in the API payload into the database. Because the email is deployed prior to writing to the database, all of the data needed to correctly populate and send the email message must be contained within the API payload.

This endpoint requires authentication using OAuth 2.0, and supports JSON and XML messages.

The URLs for this endpoint are:

- **North America**: https://api.eccmp.com/services2/api/EmailCampaign/Trigger
- **Europe**: https://api.ccmp.eu/services2/api/EmailCampaign/Trigger
- **Japan**: https://api.marketingsuite.jp/services2/api/EmailCampaign/Trigger

The following diagram depicts the basic processing flow for the EMAIL CAMPAIGN TRIGGER endpoint.



Pre-requisites

The EMAIL CAMPAIGN TRIGGER endpoint requires that the following assets be defined within your Messaging account:

- **Data Map** -- The Data Map provides data handling instructions, such as where to store the inbound data contained within the API request message, whether this data should be used to update existing records and / or create new records, and any optional formatting or special processing to perform on the inbound data. The Data Map can be created within the Messaging application, or through the use of the data map endpoint.
- **API Post** -- The API Post defines all the expected fields in the request message. The API Post must be created within the Messaging application. Please note that the API Post must have the "REST API Only" and "Triggering" options both checked, and the API Post must be published.
- **Web Authentication**: The API Post must specify an "Authentication" field, which is defined by creating a Web Authentication in the platform. This Authentication field is used during the API submission process in order to do a quick lookup to the database. The system will match to the database on this Authentication field, in order to see if this record already exists in the database; if so, the system will identify the Primary Key ID ("pk_id") of that existing record. The pk_id is needed when the data in the API payload is later loaded to the database, so that the system can update the correct record. In most cases, the Authentication field is the Unique Identifier for the Campaign source table.

- **Campaign** -- You must define and launch an Event-triggered Campaign that uses the above API Post as the event trigger.

Methods

The EMAIL CAMPAIGN TRIGGER endpoint supports the following HTTP method:

- **POST**: Trigger the deployment of an Event-triggered Campaign using the data provided in the request message to build and send the email message.

Authentication

Access to the EMAIL CAMPAIGN TRIGGER endpoint requires that you first be authenticated within the platform. Within Messaging, authentication is handled by OAuth 2.0. To authenticate with OAuth 2.0, you must first obtain a "Consumer Key" and a "Consumer Secret." Both of these values are managed at the user level, and can be obtained from within the Messaging application.

Next, you'll use your Consumer Key and Consumer Secret to request a "token." A token is a text string that, when provided in a request message, will allow the user access to the requested service. Tokens are valid only for a certain period of time.

For more details on how to authenticate your API request, please see the *Messaging: API How-to Guide*.

2 Trigger a Campaign

Overview

This section describes how to trigger the deployment of an Event-triggered Campaign via a POST request to the EMAIL CAMPAIGN TRIGGER endpoint.

This section assumes that you've already created the Event-triggered Campaign with all the necessary assets and message content, configured the Campaign with a trigger type of "API Post," and launched the Campaign.

> **Note**
>
> For general information on how to configure an email Campaign, please see the Messaging Online Help system.

Parameters

The options and parameters described in this section are used to submit data to write to the database, along with several other processing options.

recipient

This object contains all of the data needed to build and deploy the Email Campaign.

> **Note**
>
> The name "recipient" for this object can be misleading. You can use this endpoint to trigger Campaigns built off any source table, not just a "recipient" table. For example, you could use this endpoint to send order confirmations using a Campaign built off an Order table.

The parameters in this object are described below in more detail.

apiPostId

This integer parameter is required.

The **apiPostId** parameter represents the **Form ID** of an API Post. This API Post must be assigned as the trigger mechanism for the desired Event-triggered Campaign that you want to deploy.

For example:

```
"apiPostId": "2687"
```

enable_validation

This Boolean parameter is optional. If not provided, the system will default this parameter to "false."

If set to "true," the system will validate the data in the request message (see **Data Validation** below for more details), prior to loading the data to the database. If this validation fails, the data will not be written to the database, and the response message will contain the appropriate error message.

If set to "false," the system will not perform this data validation step.

Cheetah Digital recommends setting this parameter to "true."

You can use this parameter instead of, or in conjunction with, the **validate_on_error** parameter described below.

validate_on_error

This Boolean parameter is optional. If not provided, the system will default this parameter to "false."

If set to "true," the system will attempt to load all data into the database without first validating it. If the system encounters any errors when loading the data, the system will then run the data validation process (see **Data Validation** below for more details), and the response message will contain the appropriate error message.

If set to "false," the system will not perform this data validation step.

You can use this parameter instead of, or in conjunction with, the **enable_validation** parameter described above.

Cheetah Digital recommends setting this parameter to "true."

data

This array parameter is required.

The **data** array contains the data needed to populate and send the Email Campaign. The data is sent as key / value pairs consisting of a field's **Column Name** in the **name** parameter, and the corresponding value in the **value** parameter.

> **Note**
>
> The field names within the **data** array must be all lowercase.

For example:

```
"data":[
    {
      "name": "order_id",
      "value": "5150"
    },
    {
      "name": "email",
      "value": "jdoe@cheetahdigital.com"
    },
    {
      "name": "name_first",
      "value": "John"
    },
    {
      "name": "name_last",
      "value": "Doe"
    }
  ]
```

The EMAIL CAMPAIGN TRIGGER supports a single flat data structure, meaning the API payload supports sending one record, for one table (i.e., the source table for the triggered Campaign). Any fields and values you want to use for Personalization within the email content must be included within the API payload, AND those fields must be on the Campaign's source table.

As an example, let's say you're using this endpoint to send an order confirmation email, and you're using an Order table as the source for the Campaign. Your API payload must conform to the following restrictions:

- The payload can contain data for only one order.
- The payload must include all the necessary values you want to use for Personalization within the email message content.
- The fields in the payload must exist on the Order table.

Optionally, through the Data Map in your API Post, you can use the API payload to update data in joined tables. Continuing the above example, let's say your Order table joins on "email" to a Recipient table. You want to use the information you gathered from the new order to update your customer's first name and last name values on that Recipient table. In order to do this, the first name and last name fields must exist on the Order table as well, and the Data Map must reference the join on "email" to those same fields on the Recipient table.

In this manner, the system will use the first name / last name values in the API payload to update those fields in the joined Recipient table.

Record Metadata

The **data** array described above can optionally be used to submit "record metadata" fields and values that are never inserted into the platform's tables, but instead contain additional values available for Personalization within the Campaign or in Content Blocks. These values are never saved, as they don't relate to physical columns or the relational structure.

It's important to note the distinction between metadata that's provided through the EMAIL CAMPAIGN TRIGGER message (referred to as "record metadata"), and the Metadata feature

that's available within the Messaging application (referred to as "Campaign metadata"). The metadata provided through the EMAIL CAMPAIGN TRIGGER is not defined or saved anywhere in the platform. Conversely, the Metadata feature available within the application allows you to define Metadata fields, and then assign values to Campaigns; this Metadata feature is primarily intended to categorize, or "tag" Campaigns for use in reports and Filters.

To pass record metadata values in the EMAIL CAMPAIGN TRIGGER request message, you don't need to pre-define the metadata field in Messaging. Instead, you must enter a custom Merge Symbol for the metadata field into your Campaign content, then be sure to reference that exact same Merge Symbol in your API payload.

Please note that the Merge Symbol you use for this purpose must be different than any of the existing standard fields in your database. If you use a Merge Symbol for a field that's already defined in the system, then the Campaign will personalize on that field instead, and not use the value you send over in the API payload.

To add a custom Merge Symbol into your Campaign content.

1. From the System Tray, select *Campaigns* > *Management* > *Campaigns*.
2. Either select an existing Campaign to edit, or create a new Campaign.
3. Click the "Content" chevron.
4. Place your cursor in the desired position in the Content Editor.
5. Type in the Merge Symbol that you want to use. A metadata Merge Symbol is surrounded by double-brackets, and uses the prefix "meta" followed by a colon, then the symbol name. For example: {(meta:test_meta)}. Make a note of this symbol name, as you'll need to enter this exact same name into the API payload.

Next, you need to provide the Merge Symbol name and value within your message payload, being careful to use the same name.

Example:

```
"data":[
                                                                    {
"name": "test_meta",
"value": "Record Metadata test value"
                                                                    }
]
```

The value you provide in the message payload will get inserted into the message content, at the location of the Merge Symbol. This value will not be loaded into the Messaging database.

Data Validation

If you enable either of the data validation parameters described above, the system validates the data in the request message using the following rules:

Ambiguous or duplicate data validations:

- Check for duplicates in the incoming data.
- Check if the given database field names exist in the database.

Primary Key ID / Unique Identifier (i.e., "Alternate Key") validations:

- Check if the Primary Key ID or Unique Identifier field is provided.
- Check if the provided Primary Key ID or Unique Identifier for the main table is not empty.
- Check if the provided Primary Key ID or Unique Identifier for a joined table is not empty.

Data content validations:

- For fields of data type "String," "Email," "Push Registration ID," and "LINE Contact MID," the provided value is not longer than 255 bytes.
- For fields of data type "Long String," the provided value is not longer than 8000 bytes.
- For fields of data type "Money / Decimal," the provided value is between -9223372036854775808 through 9223372036854775807.
- For fields of data type "Date / Time," the provided value is within the valid date range.
- For fields of data type "Email," the provided value has the right format.
- For fields of data type "Phone," the provided value has the right format.
- For fields of data type "Big Integer," "Facebook ID," and "Twitter," the provided value is between -9223372036854775808 through 9223372036854775807.

3 Response

This section describes the possible response messages sent back from the EMAIL CAMPAIGN TRIGGER endpoint.

Success

Upon successful completion of an EMAIL CAMPAIGN TRIGGER service request, a "success" message is returned with a response code of '200.' This message includes details of every Campaign that was triggered from the API call. The response includes the Running Campaign ID of the triggered Campaign, the Message ID (a unique identifier for every email message deployed from the platform), and the date / time the email message was sent.

> **Note**
>
> The "Running Campaign ID" is a unique identifier for a Campaign that gets generated when a Campaign is launched. This ID is different from the Campaign's "Object Reference ID," which gets generated when the Campaign is initially created.

For example:

```
"sentMessages": [
                                                                {

"campaignId": 29445,

"messageId": 30848973,

"sendTime": "2018-04-18T20:00:06.2761217Z"
                                                              },
                                                               {

"campaignId": 29448,

"messageId": 30848974,
```

```
    "sendTime": "2018-04-18T20:00:06.3229385Z"
                                                                    }
    ]
```

Errors

If Messaging encounters a problem with an EMAIL CAMPAIGN TRIGGER request message, the platform will send an "error" message with details of the problem. Below is a list of error codes and their descriptions.

Respons e Code	Error message	Description
400	Invalid Data : Recipient Missing	Email address is missing or not provided within the API call
400	Invalid Data : Recipient Empty	An empty string email was provided in the payload
400	Invalid Data : Email address {email} is not valid	Email address in is invalid or incorrect (i.e. Invalid Data exception).
400	[Mailer error Message]	This error will be returned if an email address isn't flagged as invalid by the API server but produces an error response from the MTA mailer. For example, "Mailbox name not allowed. The server response was: 5.1.3 local part too long."
400	No running campaigns associated with triggered Form/Api Post	No running campaign attached to the Form ID
400	Invalid Data : Following Campaigns are invalid:	The list of campaigns in the payload are invalid
500	Message not sent. Failed to connect to the mailer [Mailer error message]	Message not sent. Failed to connect to the mailer.

4 Sample Messages

Sample Request #1

This sample request message contains the necessary values to trigger the deployment of an Event-triggered Campaign.

JSON Payload

```
{
"recipient":                          {
                  "apiPostId": "2689",
          "enable_validation": "true",
                                                "validate_on_error": "true",
                                                                    "data":[

                                                                             {


"name": "order_id",


"value": "5150"

                                                                           },

                                                                             {


"name": "email",


"value": "jdoe@cheetahdigitalcom"

                                                                           },

                                                                             {


"name": "name_first",


"value": "John"

                                                                           },

                                                                             {
```

```
"name": "name_last",

"value": "Doe",
                                                                    },
                                                                     {

"name": "test_meta",

"value": "Record Metadata test value"
                                                                     }
                                                                      ]
                                                                     }
}
```

Sample Request #2

This sample request message contains the necessary values to trigger the deployment of an Event-triggered Campaign that uses a tokenized email address field.

JSON Payload

```
{
"recipient":                                                          {
                                                  "apiPostId": "2689",
            "nontokenized_email_address" : "jdoe@cheetahdigital.com"
                                                              "data":[

                                                                     {

"name": "email",

"value": "415625lalae67d45dc36441@cheetahdigital.com"

                                                                    },
                                                                     {

"name": "name_first",

"value": "John"
```

```
                                                                    },
                                                                     {
    "name": "name_last",
    "value": "Doe",
                                                                    },
                                                                     {
    "name": "test_meta",
    "value": "Record Metadata test value"
                                                                     }
                                                                      ]
                                                                     }
}
```

5 Appendix A -- Identifiers

Messaging uses several different types of IDs when referencing assets, such as tables, fields, folders, Filters, and so forth. This appendix describes these different types of IDs, and provides steps on how to look up the value of an ID.

Form ID

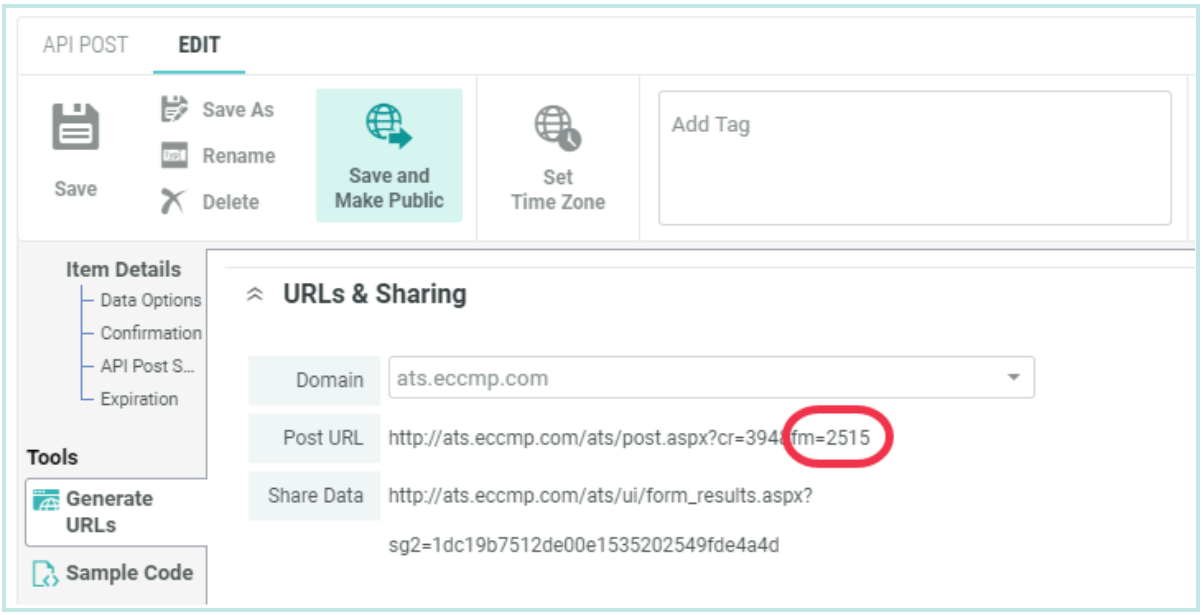
The Form ID is a system-generated identifier for every API Post in your account.

> **Note**
>
> For an API Post, the "Form ID" and the "Object Reference ID" have the same value. In common usage, when referring to the identifier for an API Post, you'll typically hear the term "Form ID" rather than "Object Reference ID."

The value for this identifier can be found within the Messaging application, or by using the SEARCH endpoint, which will return the Form ID in the response message.

To find the Form ID within the application:

1. From the System Tray, select *Data Integration* > *Processes* > *API Posts*. The system displays a list of all the API Posts in your account.
2. Select the desired API Post. The API Post Details screen is displayed.
3. From the Function Menu, select "Generate URLs." Within the "Post URL" field, the system displays the Form ID as the value of the "fm" parameter.



Optionally, you can use the SEARCH endpoint, and search for the desired API Post:

1. Submit a GET request to the SEARCH API endpoint. The simplest method is to use the versions of the SEARCH endpoint that allow you to retrieve information based on either the API Post name or its type. To retrieve information about all of your API Posts, use the asset type "ImportFormApi." For example:

```
https://api.eccmp.com/services2/api/Object?type=ImportFormApi
```

2. The response message provides a list of all the API Posts in your system that match the search criteria. Find the desired API Post in the response message.

3. As part of the API response message, the system provides the Form ID, which is referred to as the "ref_id." For example:

```
{
  "obj_id": 44737,
  "display_name": "Email Campaign Trigger API Post",
  "type_id": "ImportFormApi",
  "ref_id": 2515,
  "parent_obj_id": 37249,
  "eligibility_status_id": "READY"
}
```

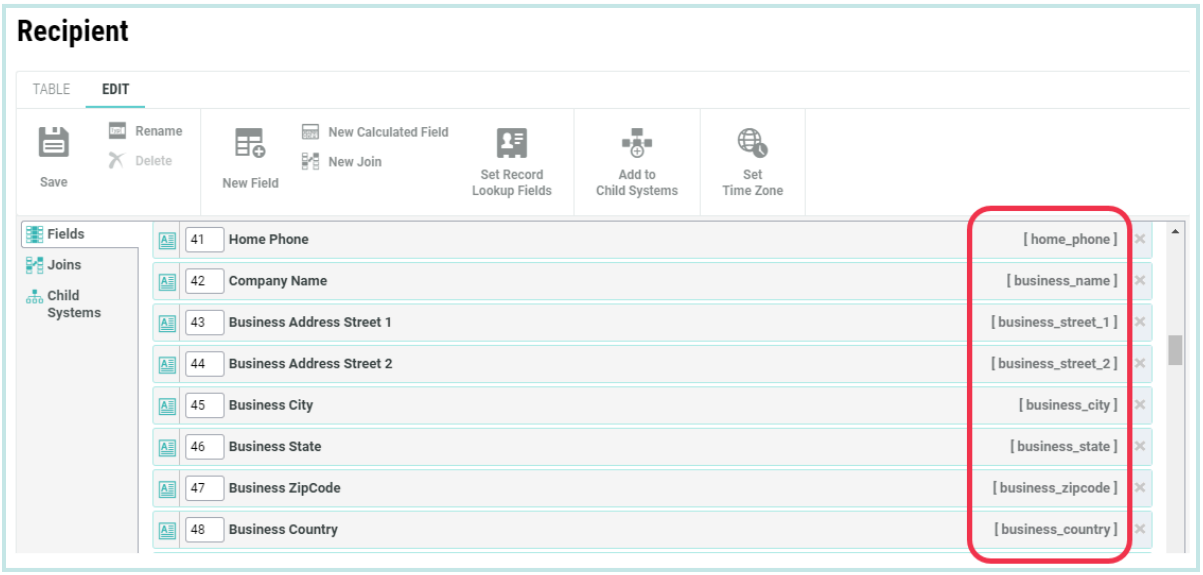
Column Name

For the field names in the **data** array, you must use the system "Column Name," and not the viewer-friendly "Display Name." The Column Names for fields can be found on the Tables screen within the Messaging application, or by using the TABLE endpoint.

> **Note**
>
> The field names within the **data** array must be all lowercase.

To look up the Column Name within the Messaging application:

1. From the System Tray, select *Data Management* > *Structures* > *Tables*. The system displays a list of all the tables in your account.
2. Select the desired table. The Table Details screen is displayed.
3. Within the list of fields in this table, the Column Name is displayed on the far-right of the screen.



To retrieve the Column Name for a field using the TABLE endpoint:

1. Submit a GET request to the TABLE API endpoint. The simplest method is to use the version of the TABLE endpoint that allows you to retrieve table information based on the table's name. For example:

```
https://api.eccmp.com/services2/api/Table?tableName=recipient
```

2. Within the API response message, the system lists every field in this table. As part of that field definition, the response includes the Column Name (referred to as the **columnName**).

Sample Response:

```
{
 "viewId": 1002,
 "entityId": 100,
 "displayName": "First Name",
 "propId": 1030,
 "columnName": "name_first"
}
```