MARIGOLD™

API Technical Guide: Email Campaign

Cheetah Digital

Table of Contents

1 Introduction

Purpose

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

Overview

The EMAIL CAMPAIGN endpoint is used to manage your Email Campaigns in Messaging, by allowing you to create new Campaigns, and to edit, view, and delete your existing Campaigns.

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
- **Europe:** https://api.ccmp.eu/services2/api/EmailCampaign
- **Japan**: https://api.marketingsuite.jp/services2/api/EmailCampaign

This same endpoint can also be used to trigger the deployment of an existing Event-triggered Campaign. When used in this manner, this trigger mechanism is called the EMAIL CAMPAIGN TRIGGER. Please see the *Messaging -- Email Campaign Trigger API Technical Guide* for more details.

Pre-requisites

When creating a new Email Campaign, the EMAIL CAMPAIGN endpoint requires that all the supporting assets needed for the Campaign must already be created and saved within

the platform. These assets can include, for example, a Filter, a Seed List, an Opt-Out Message, Content Blocks and Dynamic Blocks, and an Exclusion List. In most cases, you must reference the existing asset by means of an identifier, in order to use that asset in the Campaign.

Load and Send

The EMAIL CAMPAIGN endpoint also allows you to create, edit, and delete Load and Send Campaigns. Load and Send is an optional feature that allows you to import "campaign-ready" files for use in an Email Campaign. The term "campaign-ready" means that everything needed to build and deploy the Campaign -- including recipient contact information and any personalization fields -- are present in this file.

If you're creating a Regular One-off Load and Send Campaign, the pre-requisites (in addition to any Content Blocks, Opt-out Messages, etc.) are:

- The campaign-ready file must be successfully uploaded and processed.

If you're creating an Event-triggered Load and Send Campaign, the pre-requisites (in addition to any Content Blocks, Opt-out Messages, etc.) are:

- The campaign-ready file must be successfully uploaded and processed, or
- The folder where you intend to load the import file must be created.

Engagement Data Platform

Cheetah Messaging supports two different data sources when building Email Campaigns -- a table within the Messaging database, or the Engagement Data Platform (EDP). EDP is the data layer that powers and unifies Cheetah Digital's Messaging, Loyalty, Experience, and Personalization applications.

The EMAIL CAMPAIGN endpoints supports both data sources. You can create and manage an Email Campaign that uses either a Messaging table, or EDP, as its data source.

Unsupported Features

The EMAIL CAMPAIGN endpoint doesn't support all of the Campaign features that are available from within the Messaging application user interface. The following Email Campaign features aren't supported by the EMAIL CAMPAIGN endpoint:

- Split Cells
- A / B Testing
- Attachments
- Using a Template as the primary content source
- Custom Responses
- Quick Proofs
- Personalized URLs
- Deriving the Message Creation Start date / time by calculating backward from the Send Start date / time
- Creating or editing an Event-triggered Campaign using any trigger type other than "Advanced Event Trigger" or "File Import"
- Executing Proofs or Audits for a Load and Send Campaign.

Methods

The EMAIL CAMPAIGN endpoint supports the following HTTP methods:

- **POST**: Create a new Email Campaign.
- **GET**: Retrieve information about a specified Email Campaign.
- **PUT**: Submit modifications to an existing Email Campaign.
- **DELETE**: Delete a specified Email Campaign.

Authentication

Access to the EMAIL CAMPAIGN 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 Create an Email Campaign

Overview

This section describes how to create a new Email Campaign via a POST request to the EMAIL CAMPAIGN endpoint.

The EMAIL CAMPAIGN POST request message contains a large number of different parameters, options, and identifiers. This document presents all these parameters in a functional manner, that mimics the way the options are presented to a user in the Messaging front-end interface:

- **Campaign Setup** -- Define the Campaign type, name, location, Metadata, etc.
- **Audience** -- Define / restrict the audience of intended recipients.
- **Message** -- Define the Campaign message content.
- **Sending** -- Define the Campaign schedule.
- **Responses** -- Configure expected responses to the Campaign, such as link tracking.
- **Proofing** -- Configure proofing options.
- **Auditing** -- Configure pre-launch auditing options.

Parameters -- Campaign Setup

The parameters in this section explain the high-level options for creating and setting up an Email Campaign, such as the Campaign type, name, location, and Metadata values.

isEdpCampaignFlag

This integer parameter is optional.

The i**sEdpCampaignFlag** parameter is used to indicate that this Campaign uses EDP as its data source, rather than a Messaging table.

To indicate that this Campaign is an EDP-driven Campaign, provide a value of "1" in this parameter. If you don't provide this parameter, but you use an **entityId** value (see below) that references an EDP table, the platform will automatically poulate i**sEdpCampaignFlag** with a value of "1."

Example:

```
"isEdpCampaignFlag": 1
```

entityId

This integer parameter is required only for a Campaign that uses a Messaging table as its data source. The **entityId** parameter represents the **Entity ID** of the Campaign's source table.

If **isEdpCampaignFlag** is set to "1," any value you provide in the **entityID** parameter is ignored.

If you don't provide the i**sEdpCampaignFlag** parameter, but you use an **entityId** value that references an EDP table, the platform will automatically poulate i**sEdpCampaignFlag** with a value of "1."

If you're creating a Load and Send Campaign, this source table referenced in the **entityId** parameter must be a Load and Send table.

Example:

```
"entityId": 100
```

custId

This integer parameter is required.

The **custId** parameter represents the Customer ID of your Messaging account. The Customer ID is a unique, system-generated identifier for every Messaging client account. This value isn't displayed anywhere within the Messaging application, so you must

retrieve it by means of an API request (several different endpoints will return the Customer ID as part of the response message), or speak to your Client Services Representative, who can provide you with this value.

Example:

```
"custId": 394
```

typeId

This string parameter is required.

The **typeId** parameter indicates the type of Campaign. The valid values for this parameter are:

- "REGULAR" -- Regular One-Off Campaign
- "CALCULATED" -- Date-triggered Campaign
- "TRIGGERED" -- Event-triggered Campaign

If you're creating a Load and Send Campaign, the only valid values are "REGULAR" and "TRIGGERED;" you can't create a Date-triggered Load and Send Campaign.

Example:

```
"typeId": "REGULAR"
```

loadAndSendImportId

This integer parameter is required only for a Regular One-off Load and Send Campaign.

The **loadAndSendImportId** parameter is used to specify an Import file. You must provide the **Object Reference ID** for a specific Import file used to identify the Load and Send audience.

> If your Load and Send Campaign is an Event-triggered Campaign, you'll use the **campTriggers** object (described below) to specify the import file or folder, and not the **loadAndSendImportID** parameter.

Example:

```
    "loadAndSendImportId": "11467"
```

campTriggers

A trigger is a specific event or mechanism that causes Messaging to create and deploy a message within an Event-triggered Campaign. The **campTriggers** object is used to define the specific trigger mechanism (or mechanisms) for this Campaign.

Example:

```
"campTriggers": [
      {
                                  "typeId": "ADVANCED_EVENT_TRIGGER",

"triggerParams": []
      }
  ]
```

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

typeId

This parameter is optional.

The **typeId** parameter specifies the type of trigger mechanism. The only valid, supported values for this parameter when sending a POST message to the EMAIL CAMPAIGN endpoint are:

- "ADVANCED_EVENT_TRIGGER" -- used for Advanced Event Trigger Campaigns
- "FILE_IMPORT" -- used for Load and Send Campaigns

triggerParams

This object is used to specify additional configuration options about the trigger event.

An "ADVANCED_EVENT_TRIGGER" trigger type doesn't have any additional configuration options, but the **triggerParams** object must still be included in the message. In this case, simply provide a blank **triggerParams** object.

> **Note**
>
> If your Load and Send Campaign is a Regular One-off Campaign, you'll use the **loadAndSendImportId** parameter (described above) to specify the import file, and not the **campTriggers** object.

Example:

```
"campTriggers": [
      {
          "typeId": "FILE_IMPORT",
          "triggerParams": [
              {

                  "paramId": 1,
                  "paramName": "folder_id",
                  "integerVal": 11485

              }
          ]
      }
  ]
```

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

paramId

This integer parameter is required.

The **paramId** parameter is a sequential counter used when you're defining multiple trigger types. For a Load and Send Campaign, you can specify only one trigger, so the value of **paramId** should simply be "1."

paramName

This string parameter is required.

The **paramName** parameter is used to specify additional configuration details about the trigger. When the trigger type is "FILE_IMPORT," the valid values for **paramName** are:

- "import_id" -- Used to specify an import file
- "folder_id" -- Used to specify a folder.

integerVal

This integer parameter is required.

The **integerVal** is used to specify additional details about the trigger, based on the **paramName** value provided above.

- If **paramName** is "import_id," then you must provide the **Object Reference ID** for a specific Import file used to populate the Load and Send audience.
- If **paramName** is "folder_id," then you must provide the **Folder ID** for a folder where you will upload the Import file that will be used to populate the Load and Send audience.

obj

This object contains the name and location of the new Campaign.

Example:

```
"obj":
  {
    "display_name": "Test Email Campaign API",
    "parent_obj_id": 37249
  }
```

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

display_name

This string parameter is required.

The **display_name** parameter contains the name of the Campaign. This name must be unique within the selected folder location.

parent_obj_id

This integer parameter is optional.

The **parent_obj_id** parameter represents the **Folder ID** of the folder where you want to save the new Campaign. If you don't provide this parameter, the system will save the Campaign in the default folder location for your account, which is typically the top-most Folder in your folder structure.

campMetaParams

This object is used to assign Metadata values to the Campaign. These Metadata fields can be used to group Campaigns together, such as in Filters and reports.

Metadata fields can either be a free-form text entry field, or they can optionally contain a set of valid values from which to pick.

Example:

```
"campMetaParams": [
  {
    "optionId": 4,
    "stringVal": "Test Metadata Value"
  },
  {
    "optionId": 10,
    "selectionId": 3
  },
  {
    "optionId": 50,
    "integerVal": 25
  },
  {
    "optionId": 54,
    "datetimeVal": "2018-03-15T00:00:00"
  }
]
```

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

optionId

This integer parameter is optional.

The **optionId** references the ID of the desired Metadata field. This ID is not displayed within the user interface, and you can't search for it via an API endpoint. Please speak to your Client Services Representative who can provide this value for you.

selectionId

This integer parameter is optional.

The **selectionId** parameter is use for Metadata fields with pre-defined values; this ID references a specific pre-defined value. This ID is not displayed within the user interface, and you can't search for it via an API endpoint. Please speak to your Client Services Representative who can provide this value for you.

integerVal

This integer parameter is optional.

The **integerVal** parameter is used to provide an integer value for Metadata fields that accept Integer type data.

stringVal

This string parameter is optional.

The **stringVal** parameter is used to provide a string value for Metadata fields that accept String type data.

datetimeVal

This string parameter is optional.

The **datetimeVal** parameter is used to provide a date value for Metadata fields that accept Date type data.

Date values should be provided in the format: "YYYY-MM-YYThh:mm:ss"

Parameters -- Audience

The term "Audience" refers to the universe of recipients that you're targeting with your Campaign. The functionality differs between Campaigns that use a Messaging table as their source, and Campaigns that use EDP as their source.

For Regular One-off and Date-triggered Campaigns that use a Messaging table as their source, the Audience is defined using a Filter.

For an Event-triggered Campaign that uses a Messaging table as its source, you're not required to select an Audience Filter, as the event itself defines who the recipients are; the default Audience for an Event-triggered Campaign is "all triggered records." Optionally, however, you can select a Filter if you need to apply additional restrictions to identify and select only a sub-set of the triggered records.

For all Campaigns that use a Messaging table as their source, the Audience can optionally be restricted through the use of other assets, such as Exclusions Lists, De-duping Logic, etc.

For EDP-driven Campaigns, the Audience is defined using a Segment built in EDP; you must provide the **Segment ID** of the desired EDP Segment.

The options and parameters described in this section explain how to specify the Audience, and how to apply other optional Audience-related assets and restrictions.

toFilterId

This integer parameter is optional.

The **toFilterId** parameter represents the **Object Reference ID** of the Campaign's main audience Filter. This Filter is used to identify and select the intended recipients of the Email Campaign, for Campaigns that use a Messaging table as their source.

Please note that you can't change the Filter in a Regular One-off Campaign, once the Campaign is launched.

Example:

```
"toFilterId": 29094
```

edpSegmentId

This integer parameter is optional.

The **edpSegmentId** parameter represents the **Segment ID** of the Campaign's main audience Segment. This Segment is used to identify and select the intended recipients of the Email Campaign, for Campaigns that use EDP as their source.

Example:

```
"edpSegmentId": 5
```

ccFilterId

This integer parameter is optional.

The **ccFilterId** parameter is used to specify a Seed List. A Seed List is a group of one or more individuals who are designated to receive a copy of a Campaign message.

Seed Lists can be built in one of two ways: either by manually entering one or more individuals into the Seed List, or by using a Filter to define the logic of who should be included in the Seed List.

If using a Filter to define the Seed List, you must provide that Filter's **Object Reference ID** in the **ccFilterId** parameter.

If you're using a manually-defined Seed List, you must provide the **Object Reference ID** for that Seed List in the **ccFilterId** parameter

Example:

```
"ccFilterId": 40966
```

campPreferences

When setting up your Email Campaign, you may be required to select one or more "Preference" flags, depending on the configuration of your Sender Profile. A Preference is a special type of field in your database that's used to control whether or not a consumer has opted-in or opted-out of receiving certain types of messages from you.

The determination of whether you MUST check a Preference flag is set by a system administrator at a Sender Profile level. Each Sender Profile can optionally be configured to require a Preference flag check.

The **campPreferences** object is used to specify one or more Preference flags.

Example:

```
"campPreferences": [
  {
    "preferencePropId": 11378
  },
  {
    "preferencePropId": 11645
  }
]
```

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

preferencePropId

This integer parameter is optional.

The **preferencePropId** represents the **Field ID** for the desired Preference flag. A Campaign can optionally check multiple Preference flags.

campToList

This object is used to add additional assets to your Campaign, such as an auditing list and an Alert Group.

> **Note**
>
> The **campToList** object is also used to add a Proofing Group to your Campaign. The parameters related to Proofing are described later in this document in the **Parameters -- Proofing** section.

Example:

```
"CampToList":
  {
    "alertListId": 2803,
    "auditListId": 2070
  }
```

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

auditListId

This integer parameter is optional.

The **auditListId** is used to enable the "Delivery Audit" feature. The platform maintains a special list of email addresses across all different domains. This list functions much like a Seed List, in that your Campaign will send copies of the message to these addresses. This feature validates the entire mailing process by checking your sending infrastructure, message content, and sending reputation. The feature indicates whether your message landed in the consumer's inbox or spam folder, or if it was blocked.

To enable the Delivery Audit feature, you must provide the ID for the desired Delivery Audit list. This list is accessible only to system administrators. Please contact your Client Services Representative, who can provide you with the necessary value for this parameter.

alertListId

This integer parameter is optional.

The **alertListId** parameter is used to specify an Alert Group. The primary purpose of an Alert Group is to notify a select group of individuals when some triggering event has occurred.

To assign an Alert Group, you must provide the **Object Reference ID** for the desired Alert Group.

campToPropLists

This object is used to add Exclusion Lists to the Campaign. An Exclusion List consists of individuals who should not be targeted in your Campaign, such as individuals who work for competitors, for example. The Exclusion List automatically overrides all other criteria, including Filters, Seed Lists, and Proofing Groups. A Campaign can optionally include multiple Exclusion Lists.

If you're creating a Load and Send Campaign, the Exclusion List must be a "Virtual Exclusion List." A Virtual Exclusion List is used to synchronize opt-out requests between a client's Load and Send table and their regular relational database. At least one Virtual Exclusion List is required in a Load and Send Campaign.

Example:

```
"campToPropLists": [
    {
      "listId": 1050,
      "excludeFlag": 1
    },
    {
      "listId": 1078,
      "excludeFlag": 1
    }
],
```

The parameters in this object are described below.

listId

This integer parameter is optional.

The **listId** parameter contains the **Object Reference ID** for the desired Exclusion List.

excludeFlag

This integer parameter is optional.

The value in the **excludeFlag** parameter should be "1."

campParam

The **campParam** object contains a wide variety of different parameters. This section describes the parameters in this object that are related to Audience selection.

Example:

```
"campParam":
  {
    "ignoreConfirmationFlag": 1,
    "aetEmailBanFlag": 1,
    "aetUnsubscribeFlag": 0,
    "aet_tracking_param_flag": 1

  }
```

These parameters are described below in more detail.

ignoreConfirmationFlag

This integer parameter is optional.

In order to comply with certain regional marketing regulations, a Sender Profile can optionally be configured to enforce a "double opt-in" method for validating consumer eligibility to be contacted. If using this method, the consumer must register to receive your email messages, AND confirm the registration. Only consumers who have completed both steps in this process (i.e., the "double opt-in") will be considered eligible to receive email messages.

> **Note**
>
> Enforcement of the double opt-in process must be enabled for a Sender Profile. For more information on enabling this feature, please speak to your Client Services Representative.

If using the double opt-in process, you typically need to send a "Confirmation request" email message to a consumer who has submitted his or her initial registration. However, the consumer's status at this point in the process is still "Not confirmed" because he or she has not yet confirmed the registration. In order to deploy email messages to an "unconfirmed" consumer, the platform offers an override feature.

To enable this override for a Campaign, provide a value of "1" in the **ignoreConfirmationFlag** parameter.

aetEmailBanFlag

This integer parameter is optional.

The **aetEmailBanFlag** is relevant only for an Event-triggered Campaign that utilizes "Advanced Event Trigger" as the trigger mechanism.

> **Note**
>
> For more details on the Advanced Event Trigger feature, please see the *Messaging -- Advanced Event Trigger API Technical Guide.*

Messaging maintains a global, integrated Banned Email list that's utilized by every client in the platform. In addition, clients can optionally create their own custom Banned Email lists. When using Advanced Event Trigger to deploy email Campaigns, you have the option of validating the email addresses in the API payload against the global and custom Banned Email lists. If you enable this validation check, the system will suppress any email addresses that are found on either the global or custom lists.

To enable the Email Ban validation, provide a value of "1" in the **aetEmailBanFlag** parameter. If you don't provide this parameter, the system defaults to a value of "0."

aetUnsubscribeFlag

This integer parameter is optional.

The **aetUnsubscribeFlag** is relevant only for an Event-triggered Campaign that utilizes "Advanced Event Trigger" as the trigger mechanism.

> **Note**
>
> For more details on the Advanced Event Trigger feature, please see the *Messaging -- Advanced Event Trigger API Technical Guide.*

If you're using Advanced Event Trigger, you can check the recipient's opt-in / opt-out status in order to stay compliant with federal regulations regarding spam email. If you

enable this validation check, the system will suppress any email addresses that are found to be "opted-out."

To enable the unsubscribe validation, provide a value of "1" in the **aetUnsubscribeFlag** parameter. If you don't provide this parameter, the system defaults to a value of "0."

Please note that you can't enable **aetUnsubscribeFlag** if **aet_tracking_param_flag** is enabled. If you need to suppress unsubscribed records from your Campaign, you'll need to disable **aet_tracking_param_flag**, and then enable **aetUnsubscribeFlag**.

aet_tracking_param_flag

This integer parameter is optional.

The **aet_tracking_param_flag** is relevant only for an Event-triggered Campaign that utilizes "Advanced Event Trigger" as the trigger mechanism.

> **Note**
>
> For more details on the Advanced Event Trigger feature, please see the *Messaging -- Advanced Event Trigger API Technical Guide.*

Using AET Tracking Parameters provides increased throughput and more consistent response times. To enable AET Tracking Parameters, provide a value of "1" in the **aet_tracking_param_flag** parameter. If you don't provide this parameter, the system defaults to a value of "1."

Please note that you can't enable **aetUnsubscribeFlag** if **aet_tracking_param_flag** is enabled. If you need to suppress unsubscribed records from your Campaing, you'll need to disable **aet_tracking_param_flag**, and then enable **aetUnsubscribeFlag**.

trackingOffFlag

This integer parameter is optional.

When the trackingOffFlag is enabled, the campaign will be sent without any tracking process, such as no link tracking, spy pixel, etc. For cells and splits or AB Tests, when the tracking is switched off at the parent campaign, it is cascaded to all child cells.

campLimit

The parameters in this object are used to apply optional restrictions to the Campaign Audience.

Example:

```
"campLimit":
  {
    "msgLimit": 5000,
    "msgPerPkLimit": 1,
    "dedupePropId": 1150,
    "dedupeFilterId": 29094,
    "dedupeSortPropId": 15972,
    "dedupeSortOrder": "ASC",
    "dedupeScopeId": 200,
  }
```

These parameters are described below in more detail.

msgLimit

This integer parameter is optional.

The **msgLimit** parameter is used to set a hard limit on the total number of messages sent out in this Campaign.

msgPerPkLimit

This integer parameter is optional.

Messaging allows you to limit the Campaign to send only one message to each unique recipient over the entire lifetime of the Campaign. The platform identifies and removes duplicates using the Unique Identifier (also referred to as the "Alternate Key"). The One Message Per ID feature is typically used for triggered Campaigns, where the same individual could theoretically qualify to receive a message more than once. Using this feature, the recipient will receive only one message from the Campaign.

The One Message Per ID feature is conceptually similar to the De-Duplication Logic feature (described below), but that feature allows you to dedupe on any single field, not just on the Unique Identifier, and to define the rules for how to pick the "winner" from among a set of duplicate records.

To enable the One Message Per ID feature, provide a value of "1" in the **msgPerPkLimit** parameter.

dedupePropId

This integer parameter is optional.

De-duplication (or "dedupe") refers to the process of identifying and removing duplicate records from your Campaign Audience, in order to ensure that recipients don't receive unwanted multiple copies of your message. The De-Duplication Logic feature allows you to select what field you want to use for identifying duplicate records, as well as the rules for picking the "winner" from among a set of duplicate records.

The De-duplication Logic feature is conceptually similar to the "One Message Per ID" feature (described above), but the De-duplication Logic feature allows you to define more complex logic, and you can dedupe on a field other than the Unique Identifier.

The **dedupePropId** references the **Field ID** for the field that you want to use to identify duplicate records. The system will perform a byte-for-byte match on the values in this field to attempt to find duplicates.

dedupeFilterId

This integer parameter is optional.

The **dedupeFilterId** parameter is part of the De-Duplication Logic feature (see **dedupePropId** above for more details on this feature).

Optionally, you can use a Filter to define the "winner" from among a set of duplicate records. For example, you could define a Filter that selects records that have click activity, or that made a purchase. The **dedupeFilterId** references the **Object Reference ID** of the desired De-Duplication Logic Filter.

> **Note**
>
> If you don't define a Filter and / or Sort option to select the "winner" from among a set of duplicate records, the system will sort the duplicate set by the Primary Key ID ("pk_id") field in descending order, then pick the top-most record. This default option roughly approximates picking the "most recently added" record.

dedupeSortPropId

This integer parameter is optional.

The **dedupeSortPropId** parameter is part of the De-Duplication Logic feature (see **dedupePropId** above for more details on this feature).

Optionally, you can sort the set of duplicate records on a specified field. This option can be used with or without the Filter (see **dedupeFilterId** above for details). The system will sort the records in the duplicate set by the field you select, in the sequence you select (see **dedupeSortOrder** below), then pick the "topmost" record. For example, you could decide to pick the record with the most recent click activity, or the biggest purchase.

The **dedupeSortPropId** references the **Field ID** for the field that you want to use to sort the set of duplicate records.

dedupeSortOrder

This string parameter is optional.

The **dedupeSortOrder** parameter is part of the De-Duplication Logic feature (see **dedupePropId** above for more details on this feature).

The **dedupeSortOrder** parameter is used in conjunction with the **dedupeSortPropId** parameter described above to define the sort sequence for the specified sort field.

The valid values for **dedupeSortOrder** are:

- "ASC" -- ascending order
- "DESC" -- descending order

dedupeScopeId

This integer parameter is optional.

The **dedupeScopeId** parameter is part of the De-Duplication Logic feature (see **dedupePropId** above for more details on this feature).

For Regular One-off Campaigns, the De-Duplication Logic is always applied to the entire Campaign Audience. For Event-triggered and Date-triggered Campaigns, the **dedupeScopeId** parameter allows you to dedupe on just the records for the current batch

of recipients in the message queue, and not the entire Campaign history. The valid values for this parameter are:

- "100" -- Dedupe on just the current batch of recipients in the message queue.
- "200" -- Dedupe on the entire Campaign history.

campStepProcedures

This object is used to add custom stored procedures to your Campaign. These stored procedures must be defined and configured by your Cheetah Digital support team, which makes them available for use when setting up your Campaign.

Example:

```
"campStepProcedures": [
  {
     "seq": 1,
     "procedureId": 2
  },
  {
     "seq": 2,
     "procedureId": 3
  }
]
```

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

seq

This integer parameter is optional.

The **seq** parameter is used to define the intended sequence if you're running more than one custom stored procedure. Using this parameter, you should rank the stored procedures, starting with "1" for the first procedure, "2" for the second procedure, and so on.

procedureId

This integer parameter is optional.

The **procedureId** parameter contains the ID of the desired stored procedure.

The ID value for custom stored procedures isn't displayed within the user interface. To look up the value for this ID, you can use the CAMPAIGN PROCEDURES endpoint. This endpoint

will return a response message containing the name and ID for every custom stored procedure in your account.

Below is a sample response message from the CAMPAIGN PROCEDURES endpoint.

```
[
  {
    "procedureId": 3,
    "displayName": "Test_Coupon_Assignment"
  },
  {
    "procedureId": 2,
    "displayName": "Profile_CampaignCoupon"
  }
]
```

Parameters -- Message Content

The options and parameters in this section describe how to define the content of your email message.

contId

This integer parameter is optional.

The **contId** parameter allows you to specify an asset that you want to use as the content source for this Campaign. This parameter contains the **Object Reference ID** of the desired asset. This asset must be a Content Block or a Dynamic Block, and it must have the same source table as the Campaign.

Example:

```
"contId": 52686
```

contBodies

The email channel in Messaging allows you to define multiple "format versions" of the message content in order to accommodate the different devices and applications used by recipients to view your message. A common example is an Email Campaign that includes both an HTML version and a Plain Text version. If a recipient has an email application that's not configured to view HTML messages, he or she can still see the Plain Text version of your message.

The **contBodies** object specifies the format versions for your Email Campaign, and the message content for each format version. Within this same object, you may have one or more format versions (HTML, Plain Text, etc.), optionally with different content for each version.

Example:

```
  "contBodies":
   [
    {
      "type": "HTML",
      "usageMask": "ALL_EMAIL_STYLE_USAGE_MASK",
      "body": "<html> <body> Hello {(first_name)}!<br/><br/> Please
note that this is test HTML Content.<br/> {[Optout|46046]}</body>
</html>"
    },
    {
      "type": "TEXT",
      "usageMask": "EMAIL, WEB",
      "body": "Hello {(first_name)}! Please note that this is test
Plain Text Content. {[Optout|46046]}"
    }
   ]
```

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

type

This string parameter is optional.

The **type** parameter is used in combination with the **usageMask** parameter (described below) to define the format version of the content.

The possible values for this parameter are:

- "HTML"
- "TEXT"

usageMask

This string parameter is optional.

The **usageMask** parameter is used in combination with the **type** parameter described above to define the format version of the content.

Most of the format versions in Messaging are actually groups containing multiple sub-options. For example, the "HTML" format version contains sub-options for: Email Message, Web, Viral, Mobile Web, and iPhone. By default, all of these sub-options are contained within the parent HTML version, meaning that the same HTML content will be used for all of those different contexts.

The **usageMask** parameter allows you to pull one or more of those sub-options out of the parent version, and create a new, separate format version. This process is referred to as "promoting" a sub-option into a new group. For example, if you need the "Mobile Web" version of your HTML content to be different than the "Email Message" HTML content, you could promote "Mobile Web" to its own format version, and then provide the content unique to "Mobile Web" recipients.

This parameter can optionally contain multiple values, separated by commas.

Example:

```
"usageMask": "EMAIL, WEB_MOBILE"
```

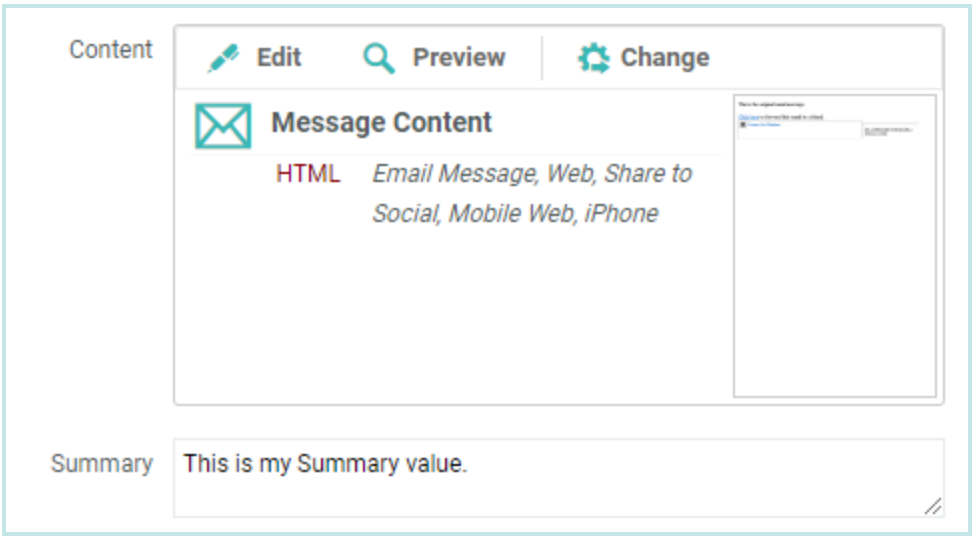
The valid values and combinations with **type**, along with the name used within the application's user interface, are listed below.

> **Note**
>
> Some of the **usageMask** values described below are designed as "bundles" of commonly used format versions.

type	usageMask	User Interface Name
HTML	EMAIL	HTML > Email Message
HTML	WEB_IPHONE	HTML > iPhone
HTML	WEB	HTML > Web
HTML	VIRAL	HTML > Viral
HTML	WEB_MOBILE	HTML > Mobile Web
HTML	ALL_EMAIL_STYLE_USAGE_MASK	Encompasses the following HTML options: Email Message, iPhone, Web, Viral, and Mobile Web.

type	usageMask	User Interface Name
HTML	ALL_WEB_STYLE_USAGE_MASK	Encompasses the following HTML options: Web, Mobile Web, and iPhone.
HTML	REPORT_SOCIAL_MASK	Encompasses the following HTML options: Viral.
HTML	REPORT_MSG_MASK	Encompasses the following HTML options: Email Message, Web, Mobile Web, and iPhone.
HTML	SUMMARY	Inserts this message content into the "Summary" field on the Campaign screen (see below for details).
HTML	NONE	Disables all HTML format versions.
HTML	ALL	All possible usage masks.
TEXT	EMAIL	Plain Text > Email Message
TEXT	WEB_SOCIAL	Plain Text > Social Text
TEXT	WEB	Plain Text > Web
TEXT	1024	Plain Text > Push Notification
TEXT	VIRAL	Plain Text > Viral
TEXT	ALL_EMAIL_STYLE_USAGE_MASK	Encompasses the following Plain Text options: Email Message, Web, Viral, and Social Text.
TEXT	ALL_WEB_STYLE_USAGE_MASK	Encompasses the following Plain Text options: Web, Social Text.
TEXT	REPORT_SOCIAL_MASK	Encompasses the following Plain Text options: Viral, Social Text.
TEXT	NONE	Disables all Plain Text format versions.
TEXT	ALL	All possible usage masks.

As described above, the "SUMMARY" **usageMask** value is used to populate the "Summary" field on the Campaign details screen.



The "Summary" field is relevant only if you're utilizing a Facebook Like Button within your message content. This Summary consists of a short message that appears on the interstitial webpage after the recipient clicks the Like button.

body

This string parameter is optional.

The **body** parameter is used to provide the actual message content for this format version, such as the HTML code or plain text.

Please note that certain characters and formatting need to be escaped, or the JSON will be considered invalid. The below characters within the HTML or Plain Text message content need to be escaped:

- **Backspace** is replaced with **\b**
- **Form feed** is replaced with **\f**
- **Newline** is replaced with **\n**
- **Carriage return** is replaced with **\r**
- **Tab** is replaced with **\t**
- **Double quote** is replaced with **\"**
- **Backslash** is replaced with ****

For assistance with escaping JSON code, the following tool will parse your JSON code and escape any problematic characters: https://www.freeformatter.com/json-escape.html.

campParam

The **campParam** object contains a wide variety of different parameters. This section describes the parameters related to message content.

Example:

```
"campParam":
  {
    "forwardProfileId": 14,
    "replyProfileId": 7,
    "aetAttachmentFlag": 1
  }
```

These parameters are described below in more detail.

forwardProfileId

This integer parameter is optional.

The **forwardProfileId** parameter is used to add a Forwarding Handler to the Campaign. Forwarding Handlers are used to automatically forward a consumer's reply (without any images or attachments), along with a standard message and subject, to a specified person or group.

To assign a Forwarding Handler, you must provide the Object Reference ID for the desired Forwarding Handler.

replyProfileId

This integer parameter is optional.

The **replyProfileId** parameter is used to add an Auto-Reply Handler to the Campaign. Auto-Reply Handlers are used to submit an automated reply message back to the recipient. The message could contain, for example, instructions on how to contact your company, a thank you for their reply, or other information.

To assign an Auto-Reply Handler, you must provide the Object Reference ID for the desired Auto-Reply Handler.

aetAttachmentFlag

This integer parameter is optional.

The **aetAttachmentFlag** is relevant only for an Event-triggered Campaign that utilizes "Advanced Event Trigger" as the trigger mechanism.

> **Note**
>
> For more details on the Advanced Event Trigger feature, please see the *Messaging -- Advanced Event Trigger API Technical Guide.*

When using AET, you have the option of sending attachments within the AET request message payload. However, in order to include those attachments within the resulting Event-triggered Campaign, you must enable this feature within the Campaign itself.

To enable the Campaign to use AET attachments, provide a value of "1" in the **aetAttachmentFlag** parameter. If you don't provide this parameter, the system defaults to a value of "0."

emailMsgTemplate

This object contains a variety of parameters related to the email header fields.

Example:

```
"emailMsgTemplate":
  {
    "fromName": "Cheetah Digital",
    "toName": "{(name_first)} {(name_last)}",
    "toAddressPropId": 1150,
    "fromAddressId": 1005,
    "codePageId": 65001,
    "subject": "Hello, {(name_first)},your monthly payment is due!"
    "bccAddressList": "jdoe@cheetahdigital.com",
    "replyToAddress": "admin@cheetahdigital.com",
    "vmtaPoolId": 100323
    "preheader": "Consider enrolling in convenient autopay"
    "preheaderPaddingFlag": 1
    "inboxSummary": "Payment is due now, enroll now in autopay for
convenience and to avoid late fees"
    }
```

These parameters are described below in more details.

fromName

This string parameter is optional.

The **fromName** parameter contains the "friendly from" value. The "friendly from" is what consumers will see in their email inbox as the email sender. In most cases, you want to use a "friendly from," such as your company name, brand name, or publication name, rather than your sending email address.

toName

This string parameter is optional.

The **toName** parameter contains the "friendly to" value. This value is displayed within the recipient's email inbox, and allows you to display something other than just the recipient's email address. By default, the platform utilizes the recipient's first name and last name as the "friendly to" value. You can also use Personalization options in this parameter by entering Merge Symbols for the desired Personalization fields.

toAddressPropId

This integer parameter is required.

The **toAddressPropId** is used to specify which field in the Campaign's source table contains the email addresses that should be used to contact recipients. You can specify any field in the source table that has a Data Type of "Email." In this parameter, provide the Field ID of the desired email address field.

fromAddressId

This integer parameter is optional.

The **fromAddressId** is used to specify the Object Reference ID of the desired "from email address."

Please note that within the EMAIL CAMPAIGN endpoint, you don't explicitly specify a Sender Profile for the Campaign; instead, you indicate the desired Sender Profile by means of the **fromAddressId**. Every Sender Profile has one or more "from email address" values assigned to it. When you specify the ID of the desired "from email address," the system

will automatically identify the Sender Profile associated to that email address, and will assign that Sender Profile to the Campaign.

codePageId

This integer parameter is optional.

The **codePageId** parameter indicates the character encoding method to use for the message content. If you don't provide a value for this parameter, the system defaults to the UTF-8 encoding method (value = "65001").

The valid values for this parameter are listed in **Appendix B**.

subject

This string parameter is optional.

The **subject** parameter contains the subject line for the email message. This value is displayed in the Subject area of the recipient's email client. You can also use Personalization options in this parameter by entering Merge Symbols for the desired Personalization fields.

bccAddressList

This string parameter is optional.

The **bccAddressList** is used to specify one or more Blind Carbon Copy (BCC) email addresses. This email address will receive a copy of the Campaign message, but this address is not visible to the recipient.

replyToAddress

This string parameter is optional.

The **replyToAddress** is used to specify one or more reply-to email addresses. This address instructs the recipient's email client where to send reply messages.

vmtaPoolId

This integer parameter is optional.

A Sender Profile can optionally have multiple VMTA Pools assigned to it. If the Sender Profile you're using for this Campaign has more than one VMTA pool associated to it, you

can specify the desired VMTA Pool by providing its Object Reference ID in the **vmtaPoolId** field.

The value for this identifier isn't displayed within the user interface, and you can't look it up via an API endpoint. If you need the ID for a VMTA Pool, please speak with your Client Services Representative, who can provide this value for you.

preheader

This string parameter is optional.

The **preheader** is used to specify the text to appear as the email preheader. This text will only appear as a preview in the mail inbox, after the subject line, but will not be included in the email. The **preheaderPaddingFlag** is set only when the **preheader** parameter contains a value.

preheaderPaddingFlag

This integer parameter is optional.

The **preheaderPaddingFlag** is used to activate the automatic smart padding added to the preheader. Smart Padding adds the perfect amount of spacing needed for each unique message to prevent the email content from immediately following the Preheader, in the inbox previews. If less padding is needed, less is used, making emails smaller and faster to load and open. Both length of subject line and length of preheader are taken into account. Padding is not added if there are no preheaders.

To enable the preheader padding, provide a value of "1" in the **preheaderPaddingFlag** parameter. To disable the preheader padding, provide a value of "0".

inboxSummary

This string parameter is optional.

The **inboxSummary** is used to specify the text to appear as the hidden summary for AI. This text will be invisible to humans and is intended for Apple, Yahoo, Google, and other email clients that scan emails to summarize them to replace your subject line or preheader with its own summary, or to insert its summary at the top of your message post-open. Enter up to 2000 characters to describe the offer or message, and the time-sensitivity of the message. In independent testing, concise explanations of up to

200 characters are most effective, however you may need a longer explanation if the email is image based without actual text.

The hiddenSummary is only used when the preheader is also used.

Parameters -- Schedule

The parameters and options described in this section are mostly related to the Campaign's schedule.

When a Campaign is launched, it goes through two separate and distinct phases: building messages and sending messages.

In the "building messages" phase, the platform identifies the intended recipients of the Campaign, identifies all the possible content variations based on the Dynamic Content options used in the Campaign, and determines the Personalization values based on any Personalization fields used in the content. The steps in this first phase are often collectively referred to as the "queue" process.

In the "sending messages" phase, the system merges together the data and the content in order to assemble the final message. This message is then transmitted to the recipients.

Each of these phases is controlled by its own dedicated schedule that defines when to start the schedule for that phase, and (for triggered Campaigns) how often to execute the process.

> **Note**
>
> For more details on the scheduling options available within the platform, please see the Online Help system, or the *Messaging -- Campaign Scheduling* document.

The EMAIL CAMPAIGN endpoint uses two separate objects to define the build schedule and the send schedule. The scheduling parameters in the POST message are largely the same within these two objects. The two objects are:

- **queueSchedule**: This object defines the start / end of the build schedule, and the build frequency. In the context of a Date-triggered Campaign, this object controls the "Recurrence Schedule."
- **sendSchedule**: This object defines the start / end of the send schedule, and the send frequency.

The above two objects should be submitted as nested objects beneath the **campParam** object.

The basic structure for defining the schedule in a Campaign is as follows:

```
"campParam":
  {
    "sendSchedule":   <define the details of the send process>
       {
         "startTime":
         "endTime":
         "timeZone":
         "dayFrequency":
            {
               <frequency details>
            }
         "timeFrequency":
            {
               <frequency details>
            }
      }
    "queueSchedule":    <define the details of the build process>
       {
         "startTime":
         "endTime":
         "timeZone":
         "dayFrequency":
            {
               <frequency details>
            }
         "timeFrequency":
            {
               <frequency details>
            }
      }
    }
```

Depending on the type of Campaign, not all of the scheduling options and parameters are relevant. As an example, a Regular One-Off Campaign builds and sends messages only once, so there's no need to define a build frequency, or a send frequency; the frequency

options are intended for triggered Campaigns which typically build and send messages multiple times.

The scheduling parameters are described below in more detail.

startTime

This string parameter is optional.

Optionally, you can use the **startTime** parameter to specify the date / time when you want the schedule to go "live." If you don't provide this parameter, the system defaults to "immediately," meaning the schedule will go live when the Campaign is launched.

Date values should be provided in the format: "YYYY-MM-DDThh:mm:ss."

Example:

```
"startTime": "2018-03-06T00:00:00"
```

endTime

This string parameter is optional.

Optionally, you can use the **endTime** parameter to define an end date / time for the schedule. If you don't provide this parameter, the system will default to running the schedule indefinitely, until the Campaign finishes, or is stopped or cancelled.

Date values should be provided in the format: "YYYY-MM-DDThh:mm:ss."

Example:

```
endTime": "2018-03-16T00:00:00"
```

timeZone

This string parameter is optional.

The **timeZone** parameter specifies the Time Zone to use for all date-related features in this Campaign. If you don't provide this parameter, the system will default to using the time zone selected in your User Profile. The valid values for this parameter are specified in Appendix B.

Example:

```
"timeZone": "Central_Standard_Time"
```

dayFrequency

The **dayFrequency** object is used to define when, and how often, the process should execute. You can define the frequency based on a daily, weekly, monthly, or yearly occurrence.

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

frequencyType

This string parameter is optional.

The **frequencyType** is used to define the unit of time for setting up the frequency. The valid values for this parameter are:

- "Daily" -- Define a daily interval at which to execute the process.
- "Weekly" -- Specify the day (or days) of the week on which to execute the process.
- "Monthly" -- Specify the day of the month on which to execute the process.
- "Yearly" -- Specify the date on which to execute the process.

Depending on which frequency type you select, the other parameters in the **dayFrequency** object will then define the details.

daysInterval

This integer parameter is optional.

If you're executing the process at a daily frequency, the **daysInterval** parameter allows you to define how many days between intervals. For example, if you want to execute the process every three days, you would provide a value of "3" in this parameter.

Example of a daily frequency:

```
"dayFrequency":
  {
    "frequencyType": "Daily",
    "daysInterval": 3
  }
```

weeklyInterval

This string parameter is optional.

If you're executing the process at a weekly frequency, the **weeklyInterval** parameter allows you to define on which days of the week you want to run the process. The valid values for this parameter are:

- "Sunday"
- "Monday"
- "Tuesday"
- "Wednesday"
- "Thursday"
- "Friday"
- "Saturday "

You can provide one or more values in this parameter; multiple values should be separated by commas.

Example of a weekly frequency:

```
"dayFrequency":
  {
    "frequencyType": "Weekly",
    "weeklyInterval": "Monday, Wednesday, Friday"
  }
```

monthlyInterval

The **monthlyInterval** object is used to define a monthly frequency.

The **monthlyInterval** object should be submitted as a nested object beneath the **dayFrequency** object.

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

intervalType

This string parameter is optional.

When setting up a monthly frequency, the system provides two different options for specifying which day in a month to run the process. The **intervalType** parameter is used to select which method to use. The valid values are:

- "DayofMonth" -- Execute process every month on a specific number ("15th of the month" for example).
- "DayType" -- Use a business rule to calculate a day on which to execute the process ("second Tuesday of every month" for example).

dayOfMonth

This integer parameter is optional.

The **dayOfMonth** parameter is used when defining a monthly frequency based on a specific day of the month. In this parameter, provide the day of the month when you want the process to execute. For example, if you want the process to run on the 15th of every month, you would provide a value of "15" in this parameter.

Example of a monthly frequency that runs on the same day every month:

```
"dayFrequency":
  {
    "frequencyType": "Monthly",
    "monthlyInterval":
      {
        "intervalType": "DayOfMonth",
        "dayOfMonth": 15
      }
  }
```

dayTypeInterval / dayType

These string parameters are optional.

The **dayTypeInterval** and **dayType** parameters are used together when defining a monthly frequency based on a business rule to calculate a date.

In the **dayTypeInterval** parameter, provide the interval for the business rule. For example, if you want the process to run on the "second Tuesday" of the month, you would indicate "second" in this parameter. The valid values for this parameter are:

- "First"

- "Second"
- "Third"
- "Fourth"
- "Fifth"

In the **dayType** parameter, indicate which day of the week, or which type of day, is needed for the business rule. For example, if you want the process to run on the "second Tuesday" of the month, you would indicate "Tuesday" in this parameter. The valid values for this parameter are:

- "Sunday"
- "Monday"
- "Tuesday"
- "Wednesday"
- "Thursday"
- "Friday"
- "Saturday "
- "WeekDay"
- "WeekendDay"
- "Day"

Example of a monthly frequency controlled by a business rule:

```
"dayFrequency":
  {
    "frequencyType": "Monthly",
    "monthlyInterval":
      {
        "intervalType": "DayType",
        "dayTypeInterval": "Second",
        "dayType": "Tuesday"
      }
  }
```

yearlyInterval

The **yearlyInterval** object is used to define a yearly frequency.

The **yearlyInterval** object should be submitted as a nested object beneath the **dayFrequency** object.

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

intervalType

This string parameter is optional.

When setting up a yearly frequency, the system provides two different options of specifying which day of the year to run the process. The **intervalType** parameter is used to select which method to use. The valid values are:

- "DayofYear" -- Execute process every year on a specific date ("January 15" for example).
- "DayType" -- Use a business rule to calculate a date on which to execute the process ("first day of July" for example).

monthOfYear / dayOfMonth

The **monthOfYear** string parameter is optional; the **dayOfMonth** integer parameter is optional.

The **monthOfYear** and **dayOfMonth** parameters are used together when defining a yearly frequency based on a specific date.

In the **monthOfYear** parameter, provide the name of the month when you want the process to execute. For example, if you want the process to run on January 15, you would provide a value of "January" in this parameter. The valid values for this parameter are:

- "January"
- "February"
- "March"
- "April"
- "May"

- "June"
- "July"
- "August"
- "September"
- "October"
- "November"
- "December"

In the **dayOfMonth** parameter, provide the day of the month when you want the process to execute. For example, if you want the process to run on January 15, you would provide a value of "15" in this parameter.

Example of a yearly frequency that runs on the same date every year:

```
"dayFrequency":
  {
   "frequencyType": "Yearly",
   "yearlyInterval":
     {
       "intervalType": "DayOfYear",
       "monthOfYear": "January",
       "dayOfMonth": 15
     }
  }
```

dayTypeInterval / dayType / monthOfYear

These three string parameters are all optional.

These parameters are used together when defining a yearly frequency based on a business rule.

In the **dayTypeInterval** parameter, provide the interval for the business rule. For example, if you want the process to run on the "first day of July," you would indicate "first" in this parameter. The valid values for this parameter are:

- "First"
- "Second"
- "Third"

- "Fourth"
- "Fifth"

In the **dayType** parameter, indicate which day of the week, or which type of day, needed for the business rule. For example, if you want the process to run on the "first day of July," you would indicate "day" in this parameter. The valid values for this parameter are:

- "Sunday"
- "Monday"
- "Tuesday"
- "Wednesday"
- "Thursday"
- "Friday"
- "Saturday "
- "WeekDay"
- "WeekendDay"
- "Day"

In the **monthOfYear** parameter, provide the name of the month when you want the process to execute. For example, if you want the process to run on the "first day of July," you would provide a value of "July" in this parameter. The valid values for this parameter are:

- "January"
- "February"
- "March"
- "April"
- "May"
- "June"
- "July"

- "August"
- "September"
- "October"
- "November"
- "December"

Example of a yearly frequency controlled by a business rule:

```
"dayFrequency":
  {
    "frequencyType": "Yearly",
    "yearlyInterval":
      {
        "intervalType": "DayType",
        "monthOfYear": "July",
        "dayType": "Day",
        "dayTypeInterval": "First"
      }
  }
```

timeFrequency

This **timeFrequency** object is used to control at what time of day, or how many times a day, the process should execute.

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

timeIntervalType

This string parameter is optional.

When setting up the "time of day" frequency, the system provides two different options for specifying at what time, or how often, to run the process. The **timeIntervalType** parameter is used to select which method to use. The valid values are:

- "OnceADay" -- Execute the process at a specified time of day.
- "MultipleTimesADay" -- Execute the process periodically throughout the day, optionally with the use of a "window" during which time the system will run the process.

runAtTime

This integer parameter is optional.

The **runAtTime** parameter is used to specify a time of day at which to execute the process. The specified time should either be on the hour, or on the half-hour. For example, 1:00, 1:30, 2:00, 2:30, and so forth.

Date values should be provided in the format: "YYYY-MM-DDThh:mm:ss." For the date component of this value, simply use "2000-01-01."

Example of a daily frequency that runs at a specified time of day:

```
"timeFrequency":
  {
    "timeIntervalType": "OnceADay",
    "runAtTime": "2000-01-01T09:00:00"
  }
```

multipleTimesInterval

This **multipleTimesInterval** object is used when you want to execute the process multiple times throughout a day, optionally with the use of a "window" during which time the system will run the process.

The **multipleTimesInterval** object should be submitted as a nested object beneath the **timeFrequency** object.

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

runIntervalUnit / runInterval

The **runIntervalUnit** string parameter is optional; the **runInterval** integer parameter is optional.

These parameters are used together when defining a frequency to execute the process multiple times a day.

The **runIntervalUnit** parameter controls the unit of measure for the frequency interval. For example, if you want the process to run every 3 hours, you would indicate "hour" as the desired unit. The valid values for this parameter are:

- "Minute"

- "Hour"

The **runInterval** parameter is used to define the frequency interval, for how often you want the process to run. For example, if you want the process to run every 3 hours, you would provide a value of "3" in this parameter.

If the **runIntervalUnit** value is "Minute," then the valid values for **runInterval** are: "10," "20," "30," "40," and "50."

If the **runIntervalUnit** value is "Hour," then the valid values for **runInterval** are the integers "1" through "11."

excludeTimeBefore / excludeTimeAfter

These string parameters are optional.

These two parameters are used to define a window during which time the process will execute. For example, let's say you want the process to run only during the normal business hours of 8:00 AM to 5:00 PM. You would use these parameters to instruct the system to exclude all frequency intervals before 8:00 AM, and after 5:00 PM.

Date values should be provided in the format: "YYYY-MM-DDThh:mm:ss." For the date component of this value, simply use "2000-01-01."

Example of a daily frequency that runs every three hours, but only during a specified processing window:

```
"timeFrequency":
  {
    "timeIntervalType": "MultipleTimesADay",
    "multipleTimesInterval":
       {
         "runIntervalUnit": "Hour",
         "runInterval": 3,
         "excludeTimeBefore": "2000-01-00T08:00:00",
         "excludeTimeAfter": "2000-01-00T17:00:00"
       }
  }
```

carryOverToNextDayFlag / stopNightDeliveryFlag

These two integer parameters are optional. These parameters should be submitted within the **campParam** object.

The **carryOverToNextDayFlag** and **stopNightDeliveryFlag** parameters are used to control the platform's Stop Night Delivery (SND) feature. SND allows you to automatically suspend message delivery during the night, and then optionally resume again in the morning.

> **Note**
> op Night Delivery must be enabled for a Sender Profile. For more information on enabling this feature, please speak to your Client Services Representative.

To enable SND for a Campaign, provide a value of "1" in the **stopNightDeliveryFlag** parameter.

Once enabled, you can use the **carryOverToNextDayFlag** to determine how you want to handle messages that don't get deployed by the end of the day. In this parameter, provide one of the following values:

- "1" -- Resume sending all unsent messages at the beginning of the next sending window.
- "0" -- Delete all unsent messages.

Example:

```
"campParam":
  {
    "stopNightDeliveryFlag": 1,
    "carryOverToNextDayFlag": 1
  )
```

stoTypeId

This string parameter should be submitted within the **campParam** object. Send Time Optimization (STO) allows you to derive the best time to contact each recipient within the Campaign's audience. The platform will deploy messages to each recipient based on their preferred time of day.

> Note: ML STO must be enabled to use this parameter. For more information on enabling this feature, please speak to your Client Services Representative.

STO is not applicable for:

- Event triggered campaign
- EDP campaign
- Non-email campaign

The **stoTypeId** parameter indicates the type of STO used in the Campaign. The valid values for this parameter are:

- "MACHINE_LEARNING" -- ML STO utilizes machine learning models based on click and open data to determine the optimal delivery time based on recipient behavior and email engagement.
- "RECIPIENT_PREFERRED_TIME" -- utilizes either a recipient's local time (as determined by Postal Code) or a preferred delivery time (based on data imported into the platform from a third-party source) to optimize Campaign delivery.
- "RECIPIENT_POSTAL_CODE" -- uses a postal code field to optimize Campaign deployment based on each recipient's local time. To use this feature, the Campaign's source table must have a dedicated Postal Code field that stores the Postal Code for each recipient in standard U.S. 5-digit ZIP Code format (not ZIP +4).
- “TIME_ZONE” -- lets you optimize Campaign deployment based on each recipient's local time zone. To use this feature, the Campaign's source table must have a dedicated Time Zone field that stores the time zone value for each recipient.
- “NONE” - useful to turn off STO from any STO type in a campaign

Based on the STO type selected, other related parameters are needed. The related compulsory parameters are needed for the launch of the Campaign.

STO Type	Related Parameters	Description
MACHINE_LEARNING	stoSettingId (mandatory)	Options: STO_HOUR / STO_AM / STO_PM.

		Use "STO Hour" to send the campaign at the best hour of the day for each recipient. STO AM hours run from 01:00 to 12:00 STO PM hours run from 13:00 to 24:00
	stoDefaultTime (optional)	Datetime value
RECIPIENT_PREFERRED_TIME	stoTimePropId (mandatory)	Datetime field from the campaign entity table
	stoPostalCodePropId (Optional)	Postal code field from the campaign entity table
RECIPIENT_POSTAL_CODE	stoPostalCodePropId (mandatory)	Postal code field from the campaign entity table
TIME_ZONE	stoTimeZonePropId (mandatory)	Postal code field from the campaign entity table
NONE	-	All other related STO param will be deleted when STO Type is changed to NONE through API

Example:

```
"campParam": {
        ...
         "stoTypeId": "RECIPIENT_PREFERRED_TIME",
         "stoTimePropId": 1519
    },
```

campLimit

Most of the parameters in the **campLimit** object are used to restrict the Campaign Audience. The following parameter relates to sending, and is described below in more detail.

Example:

```
"campLimit":
```

```
{
  "msgPerHourLimit": 3000
}
```

msgPerHourLimit

This integer parameter is optional.

By default, the platform will send messages as quickly as it can after they've been created. However, to build and maintain a good sending reputation, care must be taken to limit the number of messages being sent out from any given IP Address at one time (this process is known as "throttling"). Throttling is especially important when starting to send email messages from a new IP address (known as "IP warming").

The **msgPerHourLimit** parameter is used to set a limit on how many messages are deployed per hour. The system will round up the value you provide to the next highest increment of 500.

Parameters -- Responses

The parameters described in this section are used to configure what responses you expect to receive from your Campaign message. This response might be a link click, or an SMS text message, for example.

linkTrackingUsageMask

This string parameter is optional.

The **linkTrackingUsageMask** parameter is used to control the specific link tracking details, such as which links to track. The platform will parse your Campaign message content to attempt to identify links within the specified format version. The valid values for this parameter are:

- "NONE" -- Do not track links.
- "HTML" -- Track links in the HTML format versions only.
- "TEXT" -- Track links in the Plain Text format versions only.
- "HTML_AND_TEXT" -- Track links in both HTML and Plain Text format versions.

Example:

```
"linkTrackingUsageMask": "HTML_AND_TEXT"
```

linkTrackingDomainId

This string parameter is optional.

The **linkTrackingDomainId** parameter represents the ID of a link tracking domain. If you have multiple domains set up for your account, the **linkTrackingDomainId** is used to specify the desired domain.

The value for this identifier isn't displayed within the user interface, and you can't look it up via an API endpoint. If you need the ID for a link tracking domain, please speak with your Client Services Representative, who can provide this value for you.

Example:

```
"linkTrackingDomainId": "1506"
```

campParam

This object contains a variety of different parameters. The parameters related to responses are described below in more detail.

shortenLinksFlag

The **shortenLinksFlag** parameter is not used in Email Campaigns. This parameter is used to enable the platform's URL link shortener, which is typically used only in SMS Text Campaigns that have strict character limits.

linkRedirectUrlSuffix

This string parameter is optional.

The **linkRedirectUrlSuffix** parameter is used to provide append codes that you want appended to the tracked links in your message content. Append Codes are an optional feature used for web tracking purposes through your own tracking system, or through third-party vendors. The code (or codes) provided in this parameter will be appended to each tracked link in the campaign.

Example:

```
"campParam":
  {
    "linkRedirectUrlSuffix": "trackingcode"
  )
```

smsResponseTemplates

This object is used to define an automated SMS Keyword response action. Keywords are special words or phrases that, when detected within a recipient's SMS text message, can be used to either:

- Trigger an automated text message response back to the recipient.
- Capture the data in the recipient's text message, and store it in your database.

The **smsResponseTemplates** object is used for both of the above actions, although the relevant parameters vary slightly.

Example of an automated text message action:

```
"smsResponseTemplates":
  [
    {
      "senderId": 1,
      "groupId": 701,
      "matchPhonePropId": 11139,
      "confirmationMessage": "Thanks for your message!"
    }
  ]
```

Example of a data capture action:

```
"smsResponseTemplates":
  [
    {
      "senderId": 1,
      "groupId": 646,
      "matchPhonePropId": 11139,
      "responsePropId": 10994
    }
```

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

senderId

This integer parameter is optional.

The **senderId** parameter references the ID of the desired Short Code (Short Codes are associated with Sender Profiles). This Short Code is the number to which recipients will text their responses.

The value for this identifier isn't displayed within the user interface, and you can't look it up via an API endpoint. If you need the ID for a Short Code, please speak with your Client Services Representative, who can provide this value for you.

groupId

This integer parameter is optional.

The **groupId** refers to the **Object Reference ID** of the desired SMS Keyword Group. Keywords can be organized into groups, which allows you to link the same triggered activity to all of the words or phrases contained within that group. By using Keyword Groups, you can create a list of similar words or spelling variations.

matchPhonePropId

This integer parameter is optional.

The **matchPhonePropId** parameter references the **Field ID** of a "Phone" field on the Campaign's source table. The system will look for a match between the mobile phone number on the recipient's response and the field you specify in this parameter, in order to identify the recipient.

confirmationMessage

This string parameter is optional, and is used if using the recipient's SMS text message to trigger an automated response back to the recipient.

The **confirmationMessage** parameter contains the content of the automated text message response. This message will be sent from the platform back to the recipient, when the recipient texts one of the keywords in the specified Keyword Group.

responsePropId

This string parameter is optional, and is used if you're capturing data from the recipient's SMS text messages, and storing it in your database.

The **responsePropId** parameter references the **Field ID** on the Campaign's source table where you want to store the data you're capturing in the text message.

Parameters -- Proofing

The parameters in this section are used to configure the Campaign's proofing options. A proof is a sample message that gets sent to a select individual, or group of individuals, in order to verify that the content, format, and appearance of the message is accurate.

proofFilterId

This integer parameter is optional.

By default, the platform will use the main Campaign Audience to select and populate proofing records. However, you can optionally designate an alternate Audience from which to select the records that should be used for generating proofs. This alternate Audience is defined through the use of a Filter.

The **proofFilterId** parameter represents the **Object Reference ID** of the Proofing Filter.

Example:

```
"proofFilterId": 29094
```

campToList

The **campToList** object is used to add additional assets to your Campaign.

The parameters in this object related to Proofing are described below in more detail.

testListId

This integer parameter is optional.

The **testListId** is used to specify a Proofing Group. A Proofing Group consists of one or more individuals who will receive a test message, or "proof" prior to the Campaign being launched.

To assign a Proofing Group, you must provide the **Object Reference ID** for the desired Proofing Group.

Example:

```
"campToList":
  {
    "testListId": 2442
  )
```

campLimit

Most of the parameters in this object are used to apply optional restrictions to the Campaign Audience. The parameters related to Proofing are described below in more detail.

msgLimitTest

This integer parameter is optional.

The **msgLimitTest** parameter is used to specify the maximum number of proofing messages that you want to generate.

Example:

```
"campLimit":
  {
    "msgLimitTest": 25
  }
```

emailMsgTemplate

This object contains a variety of parameters related to the email header fields. The parameters related to proofing are described below in more details.

proofSubjectPrefix

This string parameter is optional.

The **proofSubjectPrefix** parameter is used to provide a value that will be inserted in front of the subject line of the proofing email message, in order to help the recipient identify the message as being a test proof.

Example:

```
"emailMsgTemplate":
  {
    "proofSubjectPrefix": "PROOF"
  )
```

Parameters -- Auditing

The options and parameters in this section control the pre-launch audit options. The pre-launch audit step is intended as a final check, prior to launching the Campaign.

campStatProps

This object is used to define optional Cross Tab Reports that you want to generate as part of the pre-launch audit step. To create a Cross Tab, you must specify the desired field (or fields) on the Campaign source table. The system will then generate a Cross Tab Report on that field. The Cross Tab Report displays all the unique values stored in this field, along with the number of records that contain each unique value.

Example:

```
"campStatProps": [
    {
        "propId": 15078
    },
    {
        "propId": 15972
    }
]
```

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

propId

This integer parameter is optional.

The p**ropId** parameter references the **Field ID** of the desired field on which you want to generate the Cross Tab Report.

campReviewFlags

This object is used to set optional audit "check points" during the Campaign's launch. At each indicated step, the platform will stop and wait for approval before continuing on to the next step. The possible audit check points are:

- **Queuing Statistics**: Confirm the audience counts before letting content calculation begin.

- **Content Permutations**: Review content permutations before letting personalization begin.
- **Sending**: Require approval of the overall Campaign before sending messages.

Example:

```
"campReviewFlags":
   {
     "contCalculationFlag": 0,
     "personalizationFlag": 0,
     "sendingFlag": 1
   }
```

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

contCalculationFlag

This integer parameter is optional.

The **contCalculationFlag** parameter controls the "Queuing Statistics" check point. To enable this audit check point, provide a value of "1" in this parameter. If you don't provide this parameter, the system defaults it to "0."

personalizationFlag

This integer parameter is optional.

The **personalizationFlag** parameter controls the "Content Permutations" check point. To enable this audit check point, provide a value of "1" in this parameter. If you don't provide this parameter, the system defaults it to "0."

sendingFlag

This integer parameter is optional.

The **sendingFlag** parameter controls the "Sending" check point. To enable this audit check point, provide a value of "1" in this parameter. If you don't provide this parameter, the system defaults it to "1."

3 Edit an Email Campaign

Overview

This section describes how to work with existing Email Campaigns via a GET, PUT, or DELETE request to the EMAIL CAMPAIGN endpoint.

Retrieve an Email Campaign

The GET method is used to retrieve all of the information about a specified Email Campaign.

If the Campaign is an Event-triggered Campaign, the **campTriggers** parameter lists the trigger type (or possibly multiple trigger types). The GET method will return any of the valid Trigger Types supported by the platform. However, please note that the POST and PUT methods support only one two valid values ("ADVANCED_EVENT_TRIGGER" and "FILE_IMPORT").

In short, you can retrieve information about any type of Event-triggered Campaign. Conversely, you can edit or create only Event-triggered Campaigns that use Advanced Event Trigger or a File Import as the trigger mechanism.

Example of a GET response:

```
"campTriggers": [
   {
     "typeId": "BOUNCE"
   }
 ]
```

When submitting a GET request to the EMAIL CAMPAIGN endpoint, the request message must include the Campaign's Object Reference ID as a query type parameter within the URL.

For example:

```
https://api.eccmp.com/services2/api/EmailCampaign?id=34567
```

Delete an Email Campaign

The DELETE method is used to delete a specified Email Campaign.

When submitting a DELETE request to the EMAIL CAMPAIGN endpoint, the request message must include the Campaign's **Object Reference ID** as a query type parameter within the URL.

For example:

```
https://api.eccmp.com/services2/api/EmailCampaign?id=34567
```

Edit an Email Campaign

The PUT method allows you to submit modifications to an existing Email Campaign. Using this method, you can change the Campaign name or other attributes, add or remove assets, change the message content, or execute various actions, such as send proofs, run pre-launch audits, or launch the Campaign.

When submitting a PUT request to the EMAIL CAMPAIGN endpoint, the request message must include the Campaign's **Object Reference ID** as a query type parameter within the URL.

For example:

```
https://api.eccmp.com/services2/api/EmailCampaign?id=34567
```

The parameters for the PUT method are the same as described in the **Create an Email Campaign** section, with the following additions, described below.

campId

This integer parameter is required.

The **campId** represents the Campaign's **Object Reference ID**. The value you provide in this parameter must match the **id** value sent within the URL.

Example:

```
"campId": 34567
```

campAction

This string parameter is optional.

The **campAction** parameter allows you to execute an action, or process, within the specified Campaign. The valid values for this parameter are:

- "SAVE" -- Save the Campaign.
- "PROOF" -- Send proofs.
- "AUDIT" -- Run Pre-Launch Audits.

> **Note**
>
> The "PROOF" and "AUDIT" actions aren't supported for Load and Send Campaigns via the EMAIL CAMPAIGN endpoint. If you need to launch proofs or run audits on a Load and Send Campaign, you must use the Messaging application.

- "LAUNCH" -- Launch the Campaign.
- "APPROVE" -- Approve the current launch step that's waiting for approval.
- "PAUSE" -- Un-approve the "Sending" step. The system immediately stops sending messages, but continues to queue any new triggered messages, and continues to collect activity data (opens, clicks, etc.) on any messages previously deployed. The status of the Campaign is changed to “Pending Approval.” The Sending step can later be re-approved in order to resume sending
- "SUSPEND" -- Suspend a launched Campaign; this action will cause the system to temporarily stop the Campaign from sending any messages until you resume it again. If you suspend a Date-triggered or an Event-triggered Campaign using this action, the platform will NOT queue any new records while the Campaign is suspended, even if the triggering event, or the Recurrence Frequency, occurs. New triggered records are essentially ignored until you later resume the Campaign.
- "SUSPEND_WITH_QUEUE" (Event-triggered Campaigns only) -- Suspend a launched Event-triggered Campaign; this action will cause the system to temporarily stop the Campaign from sending any messages until you resume it

again. Unlike the "SUSPEND" action described above, the platform will continue to queue new triggered records while the Campaign is suspended. If you then later resume the Campaign, the platform will deploy those queued records. This action is not available for Date-triggered and Regular One-off Campaigns.

- "RESUME" -- Resume a suspended Campaign; this action will cause the system to resume sending messages in a suspended Campaign. Any messages that were in the queue at the moment you suspended the Campaign will be deployed.
- "CANCEL" -- Cancel a launched Campaign; this action will cause the system to stop the Campaign from queueing or sending any more messages. Any messages currently in the queue at the moment you cancel the Campaign are removed, and the platform doesn't retain any history of the fact that these messages were "queued but not sent." The platform will continue to collect activity data (opens, clicks, etc.) on any records deployed prior to you canceling it. This action can't be undone, and the Campaign can't later be resumed.
- "CHANGE" -- Execute the Pick Up Changes process. When executing Pick Up Changes, please note the following conditions:

- You must first suspend the Campaign.
- You can't submit Campaign revisions AND execute Pick Up Changes within the same PUT message. If you need to modify your Campaign, send the desired changes in a PUT message (without the **campAction** parameter). Then, send a second PUT message with campAction equal to "SUSPEND" to suspend the Campaign (as noted above, you must suspend the Campaign before running Pick Up Changes). Then finally, send a third PUT message with **campAction** equal to "CHANGE" to execute Pick Up Changes.

4 Response

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

Success

A successful response to a POST message will generate a response code of "200," followed by the details of the new Email Campaign contained within the body of the response message.

A successful response to a GET message will generate a response code of "200," followed by the details of the specified Email Campaign contained within the body of the response message.

A successful response to a PUT message will generate a response code of "200," followed by the details of the modified Email Campaign contained within the body of the response message.

A successful response to a DELETE message will generate a response code of "204;" the body of the response message will be empty.

Errors

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

General Errors

Response Code	Error message	Description
500	Campaign and audience entities do not match	Source table for an asset (such as a Filter) does not match the source table specified for the Campaign. All assets used in a Campaign must be built off the same source table as the Campaign itself.
500	Processing of the HTTP request resulted in an exception.	For a PUT or GET request, mismatch in the Campaign's Object Reference ID. The ID used in the URL doesn't match the ID in the **campId** parameter within the body of the message.
400	An Obj with this name already exists	Duplicate Campaign name; **displayName** value must be unique within the specified folder.
400	Invalid EntityId	**EntityId** parameter is missing or invalid.
400	CampAction SUSPEND_WITH_QUEUE is only available for Event Triggered campaigns	The "SUSPEND_WITH_QUEUE" value for **campAction** is valid only for an Event-triggered Campaign.
400	Triggers are only allowed for event triggered campaigns	Campaign is Event-triggered but **campTriggers** was not defined.
400	Link Tracking Domain Id is wrong, it is no entry in the database	Client's link tracking domain has not been configured.
400	Default TimeZone is not set	API user does not have a time zone defined.
400	Unsubscribe check is not supported by AET TP campaign	To enable **aetUnsubscribeFlag**, you must disable **aet_tracking_param_flag.**
400	AET TP feature has been turned off for now for this campaign	**aet_tracking_param_flag** is set to "1," but the AET Tracking Parameter feature has not been enabled for this account.
400	Campaign has been launched already	When using a **campAction** value of "Launch," the specified Campaign has already been launched.
400	Mismatch between channel type ({0}) and content body type ({1}) detected	The channel of the Campaign is different than the channel of the selected content source.

Response Code	Error message	Description
400	Pickup change is not available until the campaign is suspended.	You must suspend a Campaign before running Pick Up Changes.
400	Filter is NOT allowed to change after launch.	You can't change the Filter in a Regular One-off Campaign once the Campaign is launched.
500	Execution Timeout Expired. The timeout period elapsed prior to completion of the operation or the server is not responding. The wait operation timed out.	Database server operation timeout.
500	A network-related or instance-specific error occurred while establishing a connection to SQL Server. The server was not found or was not accessible. Verify that the instance name is correct and that SQL Server is configured to allow remote connections.	Database server can not be reached.

Load and Send Errors

The following error messages are related to setting up a Load and Send Campaign. In order to specify that a Campaign is a Load and Send Campaign, the selected source table in the **entityId** parameter must be a Load and Send table.

Response Code	Error message	Description
400	Load and Send must be enabled	The source table for the Campaign (**entityID**) must be a Load and Send table.
400	Load and Send is only supported for EMAIL channel	Load and Send is supported only in the Email channel.
400	Load and Send is only supported for Regular and Triggered Campaigns	The value for **typeId** is "CALCULATED." Date-triggered Campaigns are not supported for Load and Send.

Respons e Code	Error message	Description
400	At least one virtual exclusion list is required	You didn't specify a Virtual Exclusion List in the **campToPropLists** parameter. At least one Virtual Exclusion List is required in a Load and Send Campaign.
400	Virtual exclusion list is required	The Exclusion List specified in the **campToPropLists** parameter is a Standard Exclusion List. A Load and Send Campaign supports only Virtual Exclusion Lists.
400	Only one trigger can be set for Load and Send entity	For an Event-triggered Load and Send Campaign, only one trigger can be specified.
400	Only FILE_IMPORT is supported for Load and Send entity	Invalid value for typeId in the **campTriggers** object.
400	TriggerParam section not set	**TriggerParams** object is missing; this object is required for an Event-triggered Load and Send Campaign.
400	ParamName should be import_id or folder_id	Invalid value for **paramName** in the **triggerParams** object.
400	IntegerVal integer value not set	**IntegerVal** parameter is missing; this parameter is required for an Event-triggered Load and Send Campaign.
400	ImportID not found	Unknown Import Object Reference ID in the **integerVal** parameter.
400	ImportID is still running or failed	Specified Import can't be used because the import is still in process, or because the import failed.
400	Folder not found	Unknown Folder ID in the **integerVal** parameter.
400	LoadAndSendImportId not set	**loadAndSendImportId** parameter is missing; this parameter is required for a Regular One-off Load and Send Campaign.
400	LoadAndSendImportId: not found	Unknown Import Object Reference ID in the **loadAndSendImportId** parameter.
400	Use Ux for Proof and Audit	The API doesn't support executing proofs or running audits for Load and Send Campaigns.

Response Code	Error message	Description
400	EntityID of Import ({0}) is different than entity id ({1}) of the Campaign	Source table of the import is different than the source table of the Campaign.

5 Sample Code

This section contains sample messages to help illustrate the structure and functionality of the EMAIL CAMPAIGN message.

POST Message #1

The following sample POST message creates a simple email Campaign with most of the default options, and without any of the more complex, sophisticated assets. This "bare bones" message is essentially the minimum required information needed to successfully create a new email Campaign via the EMAIL CAMPAIGN endpoint.

This email Campaign will be a Regular One-off Campaign, with the default scheduling options, one HTML content format version, an audience Filter, and all the necessary email header information.

JSON Payload

```
{
    "custId": 394,
    "entityId": 100,
    "typeId": "REGULAR",
    "toFilterId": 29094,
    "obj": {
        "display_name": "Test Regular One off Campaign",
        "parent_obj_id": 37249
    },
    "contBodies": [
        {
            "type": "HTML",
            "usageMask": "ALL_EMAIL_STYLE_USAGE_MASK",
            "body": "HTML content goes here."
        }
    ],
    "emailMsgTemplate": {
        "fromName": "Cheetah Digital",
        "toName": "{(name_first)} {(name_last)}",
        "toAddressPropId": 1150,
```

```
            "fromAddressId": 1005,
            "codePageId": 65001,
            "subject": "Hello, {(name_first)}!"
        }
    }
```

POST Message #2

This sample message builds on the previous example by adding a few new assets and options to the new Campaign. These additions are highlighted in red in the sample message below.

This Campaign is a Date-triggered Campaign, with a specified start date to the Recurring Schedule and the Send Schedule, and a Recurring Frequency of "daily, every day, at 9:00 AM." The Campaign also has two content format versions -- HTML and Plain Text.

JSON Payload

```
{
    "custId": 394,
    "entityId": 100,
    "typeId": "CALCULATED",
    "toFilterId": 29094,
    "obj": {
        "display_name": "Test Date Calculated Email Campaign",
        "parent_obj_id": 37249
    },
    "contBodies": [
        {
            "type": "HTML",
            "usageMask": "ALL_EMAIL_STYLE_USAGE_MASK",
            "body": "HTML content goes here."
        },
        {
            "type": "TEXT",
            "usageMask": "EMAIL, WEB, REPORT_SOCIAL_MASK",
            "body": "Plain text content goes here."
        }
    ],
    "emailMsgTemplate": {
        "fromName": "Cheetah Digital",
        "toName": "{(name_first)} {(name_last)}",
        "toAddressPropId": 1150,
        "fromAddressId": 1005,
        "codePageId": 65001,
        "subject": "Hello, {(name_first)}!"
    },
    "campParam": {
```

```
        "sendSchedule": {
            "startTime": "2018-03-19T09:00:00",
        },
        "queueSchedule": {
            "startTime": "2018-03-19T09:00:00",
            "dayFrequency": {
                "frequencyType": "Daily",
                "daysInterval": 1
            },
            "timeFrequency": {
                "runAtTime": "2000-01-01T09:00:00"
            }
        }
    }
}
```

6 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.

Entity ID

The Entity ID is a unique, system-generated identifier for every table in your database. This value is not displayed within the application user interface anywhere, so to get the Entity ID for a table, you must retrieve it by means of the TABLE API endpoint.

To retrieve the Entity ID for a Messaging table:

1. Submit a request to the TABLE API endpoint. The simplest method is to use the version of the TABLE endpoint that allows you to retrieve 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 the field details, the response message provides the Entity ID for this table.

Sample Response:

```
{
  "viewId": 1002,
  "entityId": 100,
  "displayName": "create_date",
  "propId": 1030,
  "columnName": "create_date"
}
```

For more details on the TABLE endpoint, please see the Messaging Online Help system or the *Messaging -- Table API Technical Guide*.

For EDP tables, you can also use the TABLE API endpoint. Use the method that returns all table information; this endpoint will return both Messaging and EDP tables. However, the response message doesn't explicity indicate which tables are from Messaging, and which are from EDP. If your tables have descriptive names, like "EDP Members," then this response message can be useful for identifying the Entity ID for an EDP table.


```
[
  {
    "viewId": 2444,
    "viewName": "Order",
    "entityId": 551,
    "tableName": "order"
  },
  {
    "viewId": 2445,
    "viewName": "EDP Members",
    "entityId": 552,
    "tableName": "edp_members"
  },
  {
    "viewId": 2380,
    "viewName": "Recipient",
    "entityId": 489,
    "tableName": "recipient"
  }
]
```


As another option for looking up the Entity ID for an EDP table, submit a GET request to EMAIL CAMPAIGN (as described **here**), and reference an existing EDP Campaign. The GET response will contain the Entity ID of the EDP table.

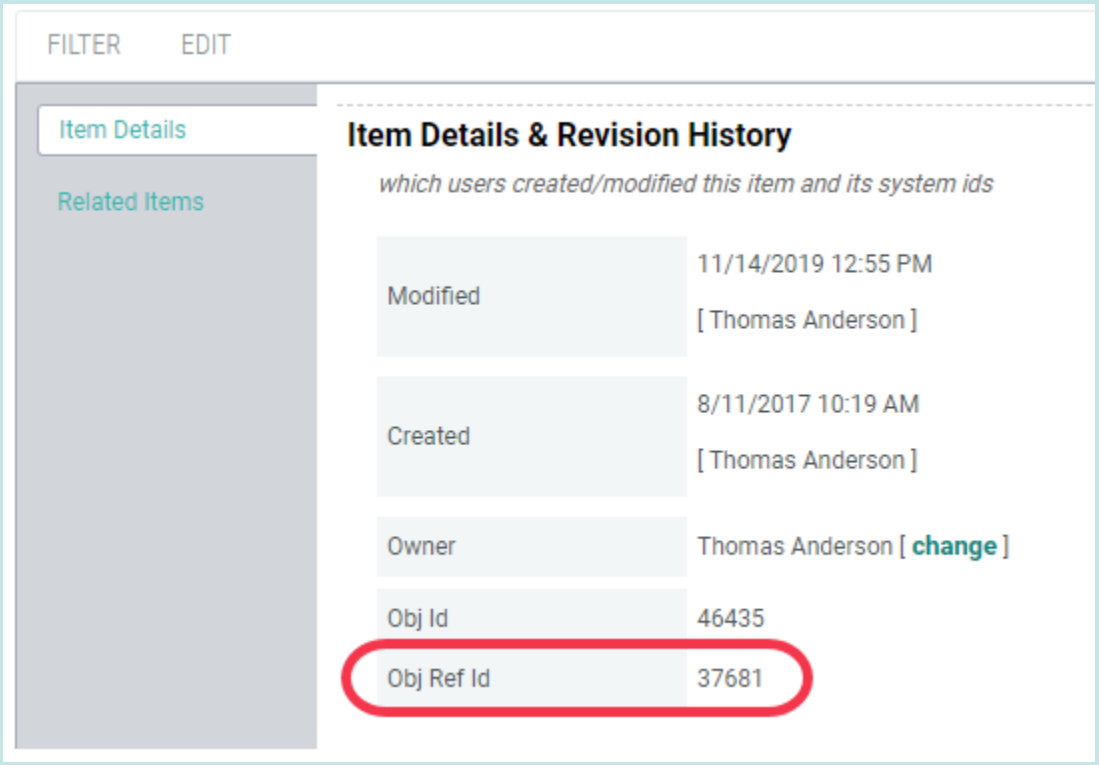
Object Reference ID

The Object Reference ID is a system-generated identifier for every item and asset in your account.

For some asset types, the value for this identifier can be found within the Messaging application:

1. From the System Tray, navigate to desired screen for this asset type.
2. In the Tool Ribbon, click the first tab; the name of this tab corresponds to the asset type, such as "Filter" if you're on the Filter screen, for example.

3. The "Item Details" screen is displayed. The Object Reference ID is listed on this screen.



Optionally, for many asset types, you can use the SEARCH endpoint, and search for the desired asset:

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 asset's name or its type. For example, to retrieve information about all of your Filters:

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

2. The response message provides a list of all the assets in your system that match the search criteria. Find the desired asset in the response message.
3. As part of the API response message, the system provides the Object Reference ID, which is referred to as the "**ref_id**." For example:

```
{
  "obj_id": 44737,
  "display_name": "Reward Members Filter",
  "type_id": "Filter",
  "ref_id": 40329,
```

```
  "parent_obj_id": 43269,
  "eligibility_status_id": "READY"
}
```

Field ID

The Field ID (or "Property ID") is a unique, system-generated identifier for every field in a table. This value is not displayed within the application user interface anywhere, so to get the Field ID for a field, you must retrieve it by means of the TABLE API endpoint.

To retrieve the Field ID for a field:

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 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 Field ID (referred to as the **propId**).

Sample Response:

```
{
 "viewId": 1002,
 "entityId": 100,
 "displayName": "create_date",
 "propId": 1030,
 "columnName": "create_date"
}
```

Folder ID

The Folder ID is a unique, system-generated identifier for each folder and sub-folder in your system. This value is not displayed within the application user interface anywhere, so to get your Folder ID, you must retrieve it by means of the SEARCH API endpoint.

1. Submit a GET request to the SEARCH endpoint. The easiest method is to use the version that lets you search by object type -- use a type value of "Folder." For example:

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

2. The response message provides a list of all the folders in your system. Find the desired folder in the response message.
3. As part of the API response message, the system provides the Folder ID, which is referred to as the "**obj_id**."

> **Note**
>
> If this Folder is a sub-folder, the "parent_obj_id" is the Folder ID of the parent folder.

Sample Response:

```
{
  "obj_id": 37465,
  "display_name": "Content Block Folder",
  "type_id": "Folder",
  "ref_id": 37465,
  "parent_obj_id": 22817,
  "eligibility_status_id": "READY"
}
```

Segment ID

For Campaigns that use EDP as their data source, the Audience for the Campaign is identified through the use of a Segment built in EDP. To designate the desired Segment, you must provide its Segment ID.

The Segment ID is a unique, system-generated identifier for each EDP Segment. This value is not displayed within the application user interface anywhere, so to get your Segment ID, you must retrieve it by means of the EDP SERVICE API endpoint.

1. Submit a GET request to the EDP SERVICE endpoint to retrieve all Segments.
2. The response message provides a list of all the Segments in your system. Find the desired Segment in the response message.
3. As part of the API response message, the system provides the Segment ID, which is referred to as the "**id**."

Sample response:


```
{
    "name": "first_name_john",
    "description": "",
    "id": 1,
    "label": "First Name John",
    "filter": "",
    "is_static": true,
    "members_included": [],
    "sql_query": "SELECT DISTINCT(MEMBERS.MEMBER_ID) FROM na394.member
AS MEMBERS   WHERE COALESCE(deactivated, false) = false AND
MEMBERS.FIRST_NAME = 'John'",
    "evaluation_path": {},
    "execution_type": "hive",
    "conditions": [],
    "members_excluded": [],
    "created_at": "2020-04-27T13:31:54Z",
    "updated_at": "2020-07-11T04:39:53Z",
    "published_at": "2020-04-27T13:40:25Z"
  }
```

7 Appendix B -- Parameter Values

This Appendix lists all of the valid values for several different parameters.

Trigger Types

When submitting a POST or PUT request message for an Event-triggered Campaign, the platform supports only one valid value for the **typeId** parameter within the **campTriggers** object: "ADVANCED_EVENT_TRIGGER." However, if you submit a GET request for an Event-triggered Campaign, the platforms supports additional values in the **typeId** parameter.

Example of a GET response:

```
"campTriggers": [
   {
     "typeId": "BOUNCE"
   }
]
```

The following table lists the possible values in the **typeId** parameter, and their description.

typeId	Description
ENTITY_UPDATE	Any Entity (i.e., Table) Updated
FILE_IMPORT	File Import
FORM_SUBMISSION	Web Form Submission
API_POST	API Post
LINK_ACTIVITY	Campaign Read
LINK_ACTIVITY_OPEN	Open HTML Activity
LINK_ACTIVITY_CLICK	Link Click
LINK_ACTIVITY_FACEBOOKLIKE	Facebook Like

typeId	Description
BOUNCE	Email Bounce
UNSUBSCRIBE	Unsubscribe
SENT_CAMPAIGN	Sent a Campaign
ADVANCED_EVENT_TRIGGER	Advanced Event Trigger
SMS_RESPONSE	Mobile (SMS) Response
WEB_EVENT_SUBMISSION	Web Event
EMAIL_SUBMISSION	Email Submission

Time Zones

The valid values for the **timeZone** parameter are as follows:

timeZone	timeZone
UTC_11	FLE_Standard_Time
Samoa_Standard_Time	Israel_Standard_Time
Hawaiian_Standard_Time	E_Europe_Standard_Time
Alaskan_Standard_Time	Arabic_Standard_Time
Pacific_Standard_Time_Mexico	Arab_Standard_Time
Pacific_Standard_Time	Russian_Standard_Time
US_Mountain_Standard_Time	E_Africa_Standard_Time
Mountain_Standard_Time_Mexico	Iran_Standard_Time
Mountain_Standard_Time	Arabian_Standard_Time
Central_America_Standard_Time	Azerbaijan_Standard_Time
Central_Standard_Time	Mauritius_Standard_Time
Central_Standard_Time_Mexico	Gegian_Standard_Time
Canada_Central_Standard_Time	Caucasus_Standard_Time
SA_Pacific_Standard_Time	Afghanistan_Standard_Time
Eastern_Standard_Time	Ekaterinburg_Standard_Time
US_Eastern_Standard_Time	Pakistan_Standard_Time

timeZone
Venezuela_Standard_Time
Paraguay_Standard_Time
Atlantic_Standard_Time
Central_Brazilian_Standard_Time
SA_Western_Standard_Time
Pacific_SA_Standard_Time
Newfoundland_Standard_Time
E_South_America_Standard_Time
Argentina_Standard_Time
SA_Eastern_Standard_Time
Greenland_Standard_Time
Montevideo_Standard_Time
UTC_02
Mid_Atlantic_Standard_Time
Azes_Standard_Time
Cape_Verde_Standard_Time
Mocco_Standard_Time
UTC
GMT_Standard_
Greenwich_Standard_Time
W_Europe_Standard_Time
Central_Europe_Standard_Time
Romance_Standard_Time
Central_European_Standard_Time
W_Central_Africa_Standard_Time
Namibia_Standard_Time
Jdan_Standard_Time
GTB_Standard_Time
Middle_East_Standard_Time

timeZone
West_Asia_Standard_Time
India_Standard_Time
Sri_Lanka_Standard_Time
Nepal_Standard_Time
Central_Asia_Standard_Time
Bangladesh_Standard_Time
N_Central_Asia_Standard_Time
Myanmar_Standard_Time
SE_Asia_Standard_Time
Nth_Asia_Standard_Time
China_Standard_Time
Nth_Asia_East_Standard_Time
Singape_Standard_Time
W_Australia_Standard_Time
Taipei_Standard_Time
Ulaanbaatar_Standard_Time
Tokyo_Standard_Time
Kea_Standard_Time
Yakutsk_Standard_Time
Cen_Australia_Standard_Time
AUS_Central_Standard_Time
E_Australia_Standard_Time
AUS_Eastern_Standard_Time
West_Pacific_Standard_Time
Tasmania_Standard_Time
Vladivostok_Standard_Time
Central_Pacific_Standard_Time
New_Zealand_Standard_Time
UTC_12

timeZone
Egypt_Standard_Time
Syria_Standard_Time
South_Africa_Standard_Time

timeZone
Fiji_Standard_Time
Kamchatka_Standard_Time
Tonga_Standard_Time

Encoding Methods

The valid values for the **codePageId** parameter are as follows:

codePageId	Encoding name	Display name
65001	utf-8	Unicode (UTF-8)
28591	iso-8859-1	Western European (ISO)
28592	iso-8859-2	Central European (ISO)
28593	iso-8859-3	Latin 3 (ISO)
28594	iso-8859-4	Baltic (ISO)
28595	iso-8859-5	Cyrillic (ISO)
28596	iso-8859-6	Arabic (ISO)
28597	iso-8859-7	Greek (ISO)
28598	iso-8859-8	Hebrew (ISO-Visual)
38598	iso-8859-8-i	Hebrew (ISO-Logical)
28599	iso-8859-9	Turkish (ISO)
28603	iso-8859-13	Estonian (ISO)
28605	iso-8859-15	Latin 9 (ISO)
50220	iso-2022-jp	Japanese (JIS)
50222	iso-2022-jp	Japanese (JIS-Allow 1 byte Kana - SO/SI)
932	shift_jis	Japanese (Shift-JIS)
50225	iso-2022-kr	Korean (ISO)
50227	x-cp50227	Chinese Simplified (ISO-2022)
1200	utf-16	Unicode