



# API Technical Guide: Push Notification Campaign

Cheetah Messaging

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
	<b>Purpose</b>	<b>7</b>
	<b>Overview</b>	<b>7</b>
	Pre-requisites	7
	<b>Unsupported Features</b>	<b>8</b>
	<b>Methods</b>	<b>8</b>
	<b>Authentication</b>	<b>8</b>
<b>2</b>	<b>Create a Push Notification Campaign</b>	<b>10</b>
	<b>Overview</b>	<b>10</b>
	<b>Parameters -- Campaign Setup</b>	<b>10</b>
	entityId	11
	custId	11
	typeId	11
	campTriggers	12
	typeId	12
	triggerParams	12
	senderProfileId	12
	obj	13
	display_name	13
	parent_obj_id	13
	campMetaParams	13
	optionId	14



selectionId	14
integerVal	14
stringVal	14
datetimeVal	15
<b>Parameters -- Audience</b>	<b>15</b>
toFilterId	15
ccFilterId	16
campToList	16
alertListId	16
campToPropLists	17
listId	17
excludeFlag	17
campLimit	17
msgLimit	18
msgPerPkLimit	18
dedupePropId	18
dedupeFilterId	19
dedupeSortPropId	19
dedupeSortOrder	20
dedupeScopeId	20
campStepProcedures	21
seq	21
procedureId	21
<b>Parameters -- Message Content</b>	<b>22</b>
contBodies	22
type	22
usageMask	23



body	23
pushNotificationMsgTemplate	23
subject	24
toAddressPropId	24
jsonPayload	24
<b>Parameters -- Schedule</b>	<b>24</b>
startTime	26
endTime	26
timeZone	27
dayFrequency	27
frequencyType	27
daysInterval	28
weeklyInterval	28
monthlyInterval	29
intervalType	29
dayOfMonth	29
dayTypeInterval / dayType	30
yearlyInterval	31
intervalType	31
monthOfYear / dayOfMonth	31
dayTypeInterval / dayType / monthOfYear	32
timeFrequency	34
timeIntervalType	34
runAtTime	35
multipleTimesInterval	35
runIntervalUnit / runInterval	35
excludeTimeBefore / excludeTimeAfter	36



<b>Parameters -- Responses</b>	<b>37</b>
linkTrackingUsageMask	37
linkTrackingDomainId	37
campParam	38
shortenLinksFlag	38
linkRedirectUrlSuffix	38
smsResponseTemplates	38
senderId	39
groupId	39
matchPhonePropId	39
confirmationMessage	40
responsePropId	40
<b>Parameters -- Proofing</b>	<b>40</b>
proofFilterId	40
campToList	40
testListId	41
campLimit	41
msgLimitTest	41
<b>Parameters -- Auditing</b>	<b>41</b>
campStatProps	42
propId	42
campReviewFlags	42
contCalculationFlag	43
personalizationFlag	43
sendingFlag	43
<b>3 Edit a Push Notification Campaign</b>	<b>44</b>
<b>Overview</b>	<b>44</b>



Retrieve a Push Notification Campaign	44
Delete a Push Notification Campaign	44
Edit a Push Notification Campaign	45
campId	45
campAction	45
<b>4 Response</b>	<b>48</b>
Success	48
Errors	48
General Errors	49
<b>5 Sample Code</b>	<b>50</b>
POST Message #1	50
POST Message #2	51
<b>6 Appendix A -- Identifiers</b>	<b>53</b>
Entity ID	53
Object Reference ID	54
Field ID	55
Folder ID	56
<b>7 Appendix B -- Parameter Values</b>	<b>57</b>
Time Zones	57



# 1 Introduction

## Purpose

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



## Overview

The **PUSH NOTIFICATION CAMPAIGN** endpoint is used to manage your Push Notification 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/PushCampaign>
- **Europe:** <https://api.ccmp.eu/services2/api/PushCampaign>
- **Japan:** <https://api.marketingsuite.jp/services2/api/PushCampaign>

## Pre-requisites

When creating a new Push Notification, the **PUSH NOTIFICATION 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, 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.

## Unsupported Features

The **PUSH NOTIFICATION CAMPAIGN** endpoint doesn't support all of the Campaign features that are available from within the Messaging application user interface. The following Push Notification Campaign features aren't supported by the **PUSH NOTIFICATION CAMPAIGN** endpoint:

- Split Cells
- Custom Responses
- 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"

## Methods

The **PUSH NOTIFICATION CAMPAIGN** endpoint supports the following HTTP methods:

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

## Authentication

Access to the **PUSH NOTIFICATION 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 a Push Notification Campaign

### Overview

This section describes how to create a new Push Notification Campaign via a POST request to the **PUSH NOTIFICATION CAMPAIGN** endpoint.

The **PUSH NOTIFICATION 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 Push Notification Campaign, such as the Campaign type, name, location, and Metadata values.



## entityId

This integer parameter is required.

The **entityId** parameter represents the **Entity ID** of the Campaign's source 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

Example:

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



## 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 for this Campaign.

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

### typeId

This string 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 **PUSH CAMPAIGN** endpoint is: "ADVANCED\_EVENT\_TRIGGER."

If you need to create an Event-triggered Push Notification Campaign using some other trigger mechanism, you'll need to create the Campaign using the Messaging application.

### 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.

Example:

```
"campTriggers": [
    {
      "typeId": "ADVANCED_EVENT_TRIGGER",
      "triggerParams": []
    }
  ]
```

### senderProfileId

This integer parameter is required.



The **senderProfileId** parameter represents the **Object Reference ID** of your Push Notification Sender Profile. A Sender Profile controls the recipient's eligibility to be contacted.

Example:

```
"senderProfileId": 69
```

## **obj**

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

Example:

```
"obj":  
{  
  "display_name": "Test Push 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.

For Regular One-off Campaigns and Date-triggered Campaigns, the Audience is defined using a Filter.

For an Event-triggered Campaign, 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 Campaign types, the Audience can optionally be restricted through the use of other assets, such as Exclusions Lists, De-duping Logic, etc.

The options and parameters described in this section explain how to specify a Filter, and 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 Push Notification Campaign.

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

Example:

```
"toFilterId": 29094
```

## 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
```

## campToList

This object is used to add additional assets to your Campaign, such as 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
}
```

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

### **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.

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."

### **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

### dedupeScopeld

This integer parameter is optional.

The **dedupeScopeld** 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 **dedupeScopeld** 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 Push Notification message.

### contBodies

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. However, unlike other channels, a Push Notification can only be Plain Text format, so you don't need to create different format versions.

The **contBodies** object specifies the Plain Text format version for your Push Notification Campaign, and the message content for it.

Example:

```
"contBodies":
  [
    {
      "type": "TEXT",
      "usageMask": "1024",
      "body": "Hello {{first_name}}!This is the Push Notification
message content."
    }
  ]
```

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.

Within a Push Notification Campaign, the only valid value for this parameter is "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.

Within a Push Notification Campaign, the only valid value for this parameter is "1024."

### **body**

This string parameter is optional.

The **body** parameter is used to provide the actual message content. Push Notifications are limited to a maximum length of 2000 characters.

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>.

### **pushNotificationMsgTemplate**

This object contains a variety of parameters related to the Push Notification Message.

Example:

```
"pushNotificationMsgTemplate":  
{  
  "Subject": "Push message subject line",  
  "ToAddressPropId": "215",  
  "JsonPayload": "string"  
}
```



These parameters are described below in more details.

### **subject**

This string parameter is optional.

The **subject** parameter contains the subject line for the Push Notification message.

### **toAddressPropId**

This integer parameter is required.

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

### **jsonPayload**

This string parameter is optional.

The **jsonPayload** parameter is used to specify the contents of the JSON payload that gets sent in the Push Notification.

## **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 **PUSH NOTIFICATION 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 15<sup>th</sup> 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"
  }
}
```

## 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.
- "TEXT" -- Track links in the Plain Text format versions only.



Example:

```
"linkTrackingUsageMask": "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 used to enable the platform's URL link shortener.

#### **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 like Omniture and Google Analytics. 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  
  }
```

## 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 **propId** 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 a Push Notification Campaign



## Overview

This section describes how to work with existing Push Notification Campaigns via a GET, PUT, or DELETE request to the **PUSH NOTIFICATION CAMPAIGN** endpoint.

## Retrieve a Push Notification Campaign

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

If the Campaign is an Event-triggered Campaign, the response message will not include any details of the trigger type or the trigger configuration.

When submitting a GET request to the **PUSH NOTIFICATION 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/PushCampaign?id=34567
```

## Delete a Push Notification Campaign

The DELETE method is used to delete a specified Push Notification Campaign.



When submitting a DELETE request to the **PUSH NOTIFICATION 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/PushCampaign?id=34567
```

## Edit a Push Notification Campaign

The PUT method allows you to submit modifications to an existing Push Notification 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 **PUSH NOTIFICATION 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/PushCampaign?id=34567
```

The parameters for the PUT method are the same as described in the **Create a Push Notification 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.
- "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 **PUSH NOTIFICATION CAMPAIGN** endpoint.



## Success

A successful response to a POST message will generate a response code of "200," followed by the details of the new Push Notification 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 Push Notification 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 Push Notification 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 **PUSH NOTIFICATION 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 <b>campId</b> parameter within the body of the message.
500	An Obj with this name already exists	Duplicate Campaign name; <b>displayName</b> value must be unique within the specified folder.
400	Invalid EntityId	<b>EntityId</b> parameter is missing or invalid.
400	CampAction SUSPEND_WITH_QUEUE is only available for Event Triggered campaigns	The "SUSPEND_WITH_QUEUE" value for <b>campAction</b> is valid only for an Event-triggered Campaign.
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.



## 5 Sample Code

This section contains sample messages to help illustrate the structure and functionality of the **PUSH NOTIFICATION CAMPAIGN** endpoint.

### POST Message #1

The following sample POST message creates a simple Push Notification 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 Push Notification Campaign via the **PUSH NOTIFICATION CAMPAIGN** endpoint.

This Push Notification Campaign will be a Regular One-off Campaign, with the default scheduling options, a text format version, and an audience Filter.

#### *JSON Payload*

```
{
  "custId": 394,
  "entityId": 100,
  "typeId": "REGULAR",
  "toFilterId": 29094,
  "senderProfileId": 69,
  "contBodies": [
    {
      "type": "TEXT",
      "usageMask": "1024",
      "body": "Push message content goes here."
    }
  ],
  "pushNotificationMsgTemplate": {
    "Subject": "This is the subject for my push message",
```



```

    "ToAddressPropId": "4545"
    },
    "obj": {
        "display_name": "Test Push Campaign",
        "parent_obj_id": 37249
    }
}

```

## 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."

### *JSON Payload*

```

{
    "custId": 394,
    "entityId": 100,
    "typeId": "CALCULATED",
    "toFilterId": 29094,
    "senderProfileId": 69,
    "contBodies": [
        {
            "type": "TEXT",
            "usageMask": "1024",
            "body": "Push message content goes here."
        }
    ],
    "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"
        }
    },
    "pushNotificationMsgTemplate": {
        "Subject": "This is the subject for my push message",
        "ToAddressPropId": "4545"
    },
    "obj": {
        "display_name": "Test Date Calculated Push Campaign",
        "parent_obj_id": 37249
    }
}

```



## 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 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*.

## 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.

The screenshot shows the 'Item Details & Revision History' screen. The sidebar on the left has 'Item Details' selected. The main content area has a title 'Item Details & Revision History' and a subtitle 'which users created/modified this item and its system ids'. Below this is a table of revision history:

Action	Date	User
Modified	11/14/2019 12:55 PM	[ Thomas Anderson ]
Created	8/11/2017 10:19 AM	[ Thomas Anderson ]
Owner		Thomas Anderson [ <a href="#">change</a> ]
Obj Id		46435
Obj Ref Id		37681



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:

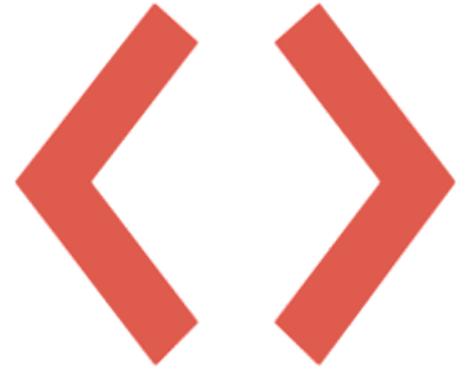
```
{
  "obj_id": 37465,
  "display_name": "Content Block Folder",
  "type_id": "Folder",
  "ref_id": 37465,
  "parent_obj_id": 22817,
  "eligibility_status_id": "READY"
}
```



}



# 7 Appendix B -- Parameter Values



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

## Time Zones

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

<b>timeZone</b>
UTC_11
Samoa_Standard_Time
Hawaiian_Standard_Time
Alaskan_Standard_Time
Pacific_Standard_Time_Mexico
Pacific_Standard_Time
US_Mountain_Standard_Time
Mountain_Standard_Time_Mexico
Mountain_Standard_Time
Central_America_Standard_Time
Central_Standard_Time
Central_Standard_Time_Mexico
Canada_Central_Standard_Time
SA_Pacific_Standard_Time
Eastern_Standard_Time
US_Eastern_Standard_Time
Venezuela_Standard_Time
Paraguay_Standard_Time

<b>timeZone</b>
FLE_Standard_Time
Israel_Standard_Time
E_Europe_Standard_Time
Arabic_Standard_Time
Arab_Standard_Time
Russian_Standard_Time
E_Africa_Standard_Time
Iran_Standard_Time
Arabian_Standard_Time
Azerbaijan_Standard_Time
Mauritius_Standard_Time
Gegian_Standard_Time
Caucasus_Standard_Time
Afghanistan_Standard_Time
Ekaterinburg_Standard_Time
Pakistan_Standard_Time
West_Asia_Standard_Time
India_Standard_Time



timeZone
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
Egypt_Standard_Time
Syria_Standard_Time

timeZone
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
Fiji_Standard_Time
Kamchatka_Standard_Time



**timeZone**

South\_Africa\_Standard\_Time

**timeZone**

Tonga\_Standard\_Time

