API Technical Guide: Folder

Cheetah Messaging

Table of Contents

1 Introduction

Purpose

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

Overview

The FOLDER endpoint is used to manage folders within your Messaging account. Once created, you can save other assets and items within the folder. You can create a folder at the top-most "root" level in your account, or as a nested folder inside of an existing folder.

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

Methods

The FOLDER endpoint supports the following HTTP methods:

- **POST**: Create a new folder.
- **GET**: Retrieve details of a folder.

- **PATCH**: Update a folder name.

Authentication

Access to the FOLDER 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 Folder

Overview

This section describes how to create a new folder via a POST request to the FOLDER endpoint.

Parameters

The options and parameters described in this section explain how to create a new folder.

folder_name

This string parameter is required.

The **folder_name** parameter represents the display name of the new folder. The folder name must be unique within the parent folder, with a maximum allowable length of 255 characters.

For example:

```
"folder_name": "Campaigns"
```

parent_folder_id

This string parameter is optional.

The **parent_folder_id** parameter represents the **Folder ID** for the parent folder of the new folder.

If you don't provide this parameter, the platform defaults to creating the new folder within the top-level "root" folder in your account.

```
"parent_folder_id": 61532
```

3 Edit a Folder

Overview

This section describes how to work with existing folders via a GET, PATCH, or DELETE request to the FOLDER endpoint.

Retrieve a Folder

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

When submitting a GET request to the FOLDER endpoint, the request message can include a Folder ID as a query type parameter within the URL. This parameter is optional.

For example:

```
https://api.eccmp.com/services2/api/Folder?folder_id=34456
```

Or

```
https://api.eccmp.com/services2/api/Folder/{34456}
```

If you submit a GET request without specifying a Folder ID, the platform defaults to retrieving information about the top-level "root" folder in your account (i.e., "My Files").

Edit a Folder

The PATCH method allows you to change the name of an existing folder.

When submitting a PATCH request to the FOLDER endpoint, the request message must include the folder's Folder ID as a query type parameter within the URL.

For example:

```
https://api.eccmp.com/services2/api/Folder?folder_id=34456
```

Or

```
https://api.eccmp.com/services2/api/Folder/{34456}
```

The additional parameters for the PATCH method are described below.

folder_name

This string parameter is required.

The **folder_name** parameter represents the modified display name of the folder. The folder name must be unique within the parent folder, with a maximum allowable length of 255 characters.

For example:

```
"folder_name": "Promo Campaigns"
```

Delete a Folder

The DELETE method is used to delete a specified folder.

> **Note**
>
> When you delete a folder, you also delete the contents of that folder.

When submitting a DELETE request to the FOLDER endpoint, the request message must include the Folder ID as a query type parameter within the URL.

For example:

```
https://api.eccmp.com/services2/api/Folder?folder_id=34456
```

Or

```
https://api.eccmp.com/services2/api/Folder/{34456}
```

4 Response

This section describes the response message sent back from the FOLDER endpoint.

Success

A successful response to a POST message to create a folder will generate a response code of "200." The response message will contain the Folder ID of the newly created Folder.

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

A successful response to a PATCH message will generate a response code of "200," followed by the modified details of the specified folder 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 a FOLDER request message, the platform will send an "error" message with details of the problem. Below is a list of error codes and their descriptions.

Response Code	Error message	Description
400	Parent folder does not exist	In a POST request, Folder ID provided in parent_folder_id is invalid or does not exist.

Response Code	Error message	Description
400	Folder Name is required	folder_name parameter is missing.
400	Folder Name cannot contain irregular characters like &, >, <, \, "	Invalid characters in the folder_name value.
400	Folder Name cannot contain more than 255 characters	folder_name value is too long.
400	A folder with that name already exists in this parent folder	Duplicate folder_name value.
400	Folder does not exist	Folder ID provided in folder_id does not exist.
400	Folder Name is required	folder_name parameter is blank or missing.
400	folder_id is missing for Folder	In a PATCH request, folder_id parameter is missing.
400	folder_id cannot be changed	In a PATCH request, you can't modify the folder's Folder ID. Don't include the folder_id within the message body.
400	parent_folder_id cannot be changed	In a PATCH request, you can't modify the folder's parent folder. Don't include the parent_folder_id within the message body.
400	The request body was empty. Please make sure that the payload is correctly formatted	Request JSON is blank or invalid.
400	Main folder cannot be deleted.	In a DELETE request, you can't delete the main "My Files" root folder in an account.
401	Authorization has been denied for this request	User is not authorized.
404	The resource cannot be found.	Non-numeric values provided in folder_id.
500	Could not find stored procedure 'p_om_obj_xxx'	Missing stored procedure

Response Code	Error message	Description
500	Execution Timeout Expired. The timeout period elapsed prior to completion of the operation or the server is not responding. System.ComponentModel.Win32Exception (0x80004005): 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.

5 Sample Messages

This section contains sample messages for the FOLDER endpoint.

Request #1

Below is a sample POST request. This request uses the **parent_folder_id** parameter to identify the parent folder of the new folder.

```
{
  folder_name: "Campaigns",
  parent_folder_id: 36472
}
```

Response #1

This sample POST response message shows the results of the above request message. The Folder ID of the new Folder can be found in the **folder_id** parameter.

```
{
  "folder_id": 37661,
  "folder_name": "Campaigns",
  "parent_folder_id": 36472,
  "child_folders": {}
}
```

Request #2

Below is a sample POST request. This request does not contain the **parent_folder_id**, so the new folder will be created in the top-level root folder in the client's account.

```
{
  folder_name: "Supporting Assets"
}
```

Response #2

This sample POST response message shows the results of the above request message. The new folder was created in the top-level root folder, which typically has a value of "11440."

```
{
                                        "folder_id": 37662,
                                "folder_name": "Supporting Assets",
                                        "parent_folder_id": 11440,
                                        "child_folders": {}
}
```

Response #3

This sample GET response message shows the results of a GET request message. The **child_folders** parameter lists all sub-folders beneath the requested folder.

```
{
                                        "folder_id": 37661,
                                "folder_name": "Campaigns",
                                 "parent_folder_id": 36472,
                                        "child_folders": {

37828: "2018 Campaigns",

37876: "2019 Campaigns",

37912: "2020 Campaigns"
                                                                        }
}
```

6 Appendix -- 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.

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"
}
```