MARIGOLD™

API Technical Guide: FTP Profile and Template

Cheetah Messaging

Table of Contents

1 Introduction

Purpose

The purpose of this document is to provide an overview of the FTP PROFILE and FTP IMPORT TEMPLATE API endpoints within the Cheetah Messaging platform. This document discusses the intended use of these two endpoints, and provides technical details for how to implement them.

Overview

The FTP PROFILE endpoint allows you to manage your FTP Profiles. FTP Profiles contain the necessary information to connect to an FTP server when transmitting files to and from Messaging. These FTP Profiles are reusable, so that they can be selected when defining various other asset types, such as FTP Import Templates, Export Templates, and Print Campaigns.

With the FTP PROFILE endpoint, you can define a new FTP Profile, and view, edit, and delete existing FTP Profiles.

The FTP IMPORT TEMPLATE endpoint allows you to manage your FTP Import Templates. These Templates allow you to set up repeatable, scheduled FTP import processes. The FPT Import Template controls the schedule of when to run the import, as well as details about the file handling settings.

With the FTP IMPORT TEMPLATE endpoint, you can define a new FTP Import Template, and view, edit, and delete existing FTP Import Templates.

Both of these endpoints require authentication using OAuth 2.0, and support JSON and XML messages.

The URLs for these endpoints are:

    - North America:
- https://api.eccmp.com/services2/api/FtpProfile
- https://api.eccmp.com/services2/api/FtpTemplate
    - Europe:
- https://api.ccmp.eu/services2/api/FtpProfile
- https://api.ccmp.eu/services2/api/FtpTemplate
    - Japan:
- https://api.marketingsuite.jp/services2/api/FtpProfile
- https://api.marketingsuite.jp/services2/api/FtpTemplate

Methods

The FTP PROFILE endpoint supports the following HTTP methods:

- **POST**: Create a new FTP Profile.
- **GET**: Retrieve information about all FTP Profiles.
- **GET**: Retrieve information about a specific FTP Profile.
- **PUT**: Edit the details of a specific FTP Profile.
- **DELETE**: Delete a specific FPT Profile.

The FTP IMPORT TEMPLATE endpoint supports the following HTTP methods:

- **POST**: Create a new FTP Import Template.
- **GET**: Retrieve information about a specific FTP Import Template.
- **PUT**: Edit the details of a specific FTP Import Template.
- **DELETE**: Delete a specific FPT Import Template.

Authentication

Access to the FTP PROFILE and FTP IMPORT TEMPLATE endpoints 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 FTP Profile

Overview

This section describes how to create a new FTP Profile using a POST request to the FTP PROFILE endpoint.

Parameters

The parameters when sending a POST request to the FTP PROFILE endpoint are described below in more detail.

profile_name

This string parameter is required.

The **profile_name** parameter contains the display name of the new FTP Profile. The name must be unique within your client account.

> **Note**
>
> If you create a new FTP Profile from within the Messaging application, the platform automatically generates the FTP Profile name for you based on a combination of the URL and the User Name. When creating an FTP Profile through the FTP PROFILE endpoint, you're allowed to define a custom name.

Example:

```
"profile_name": "Test FTP Profile"
```

user_name

This string parameter is required.

The **user_name** parameter represents the username needed to log onto the FTP Server.

Example:

```
"user_name": "CheetahAdmin"
```

password

This string parameter is required.

The **password** parameter represents the password needed to log onto the FTP Server.

Example:

```
"password": "qaCh33tah!"
```

password_confirm

This string parameter is optional.

The **password_confirm** parameter is used to confirm the password value you provided above in the **password** parameter.

Example:

```
"password_confirm": "qaCh33tah!"
```

domain_name

This string parameter is required.

The **domain_name** parameter represents the URL for the FTP Server.

Example:

```
"domain_name": "sftp://sftp.cheetah.com"
```

passive_mode_flag

This integer parameter is optional.

By default, Messaging will attempt to connect to the FTP Server in "active" mode. To instead connect in "passive" mode, provide a value of "1" in the **passive_mode_flag** parameter.

If you don't provide this parameter, the system defaults to a value of "0" (active mode).

Example:

```
"passive_mode_flag": 1
```

3 View or Edit an FTP Profile

Overview

This section describes how to work with existing FTP Profiles via GET, PUT, or DELETE requests to the FTP PROFILE endpoint.

Retrieve FTP Profiles

The GET method is used to retrieve information about all of your FTP Profiles, or about a single, specified FTP Profile. Both of these options are described below in more detail.

Retrieve All FTP Profiles

This endpoint allows you to retrieve a list of all available FTP Profiles in your account.

This method doesn't use any parameters.

Retrieve a Single FTP Profile

This endpoint allows you to retrieve information about a single FTP Profile.

The request message must include the FTP Profile's Object Reference ID. The Object Reference ID must be sent as a query type parameter within the URL.

For example:

```
https://api.eccmp.com/services2/api/FtpProfile?id=41
```

Edit an FTP Profile

The PUT method allows you to submit modifications to an existing FTP Profile. Using this method, you can change any aspect of the FTP Profile.

The request message must include the FTP Profile's **Object Reference ID** and the desired changes. The Object Reference ID must be sent as a query type parameter within the URL.

For example:

```
https://api.eccmp.com/services2/api/FtpProfile?id=41
```

All of the other FTP Profile parameters are sent within the body of the message. The parameters for the PUT method are the same as described in the **Create an FTP Profile** section, with the following additions, described below.

Please note that when submitting a PUT request, if you include the **password** parameter, you must also include the **password_confirm** parameter, and the values in these two parameters must match.

profile_id

This integer parameter is required.

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

Example:

```
"profile_id": 41
```

Delete an FTP Profile

The DELETE method allows you to delete an existing FTP Profile. Please note that deleting an FTP Profile is not supported through the Messaging user interface; the only way to delete an FTP Profile is through the FTP PROFILE endpoint.

The request message must include the FTP Profile's **Object Reference ID**. The Object Reference ID must be sent as a query type parameter within the URL.

For example:

```
https://api.eccmp.com/services2/api/FtpProfile?id=41
```

4 Create an FTP Import Template

Overview

This section describes how to create a new FTP Import Template using a POST request to the FTP IMPORT TEMPLATE endpoint.

When creating an FTP Import Template through the API, the endpoint doesn't support all of the features that are available from within the Messaging user interface. The following features aren't supported by the FTP IMPORT TEMPLATE endpoint:

- The ability to select a custom Hierarchy Rule.
- The ability to define error thresholds and automated notifications.
- The option to NOT keep detailed history of record updates; this option will be enabled by default.
- The option to NOT automatically create a Filter that selects the records in the import; this option will be enabled by default.

The FTP IMPORT TEMPLATE 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 interface:

- **Schedule** -- Define the import processing schedule.
- **FTP Download** -- Define the download configuration details.
- **File to Import** -- Define the processing details for how each file should be handled within the Messaging platform.

Parameters -- Schedule

The parameters and options described in this section are related to the FTP Import Template's processing schedule. You can optionally provide the scheduling details within the payload of the POST message (as described below), or you can reference an existing schedule using the **schedule_id** parameter (see **Parameters -- FTP Download** for more details).

scheduleControl

This object is optional.

The **scheduleControl** object is used to provide the details for the desired processing schedule for this FTP Import Template.

Instead of providing the schedule details within the payload of the API message, you can optionally use the **schedule_id** parameter to reference an existing schedule.

The parameters in the **scheduleControl** object are described below.

startTime

This string parameter is optional.

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

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

Example:

```
"startTime": "2018-03-06T01:00:00"
```

endTime

This string parameter is optional.

Optionally, you can use the **endTime** parameter to define an end date / time for the FTP Import Template processing schedule. If you don't provide this parameter, the system will default to running the schedule indefinitely, until the schedule is manually stopped.

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 FTP Import Template. 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"
- "Last"

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"
- "Last"

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 -- FTP Download

The parameters and options described in this section are related to the FTP download process.

cust_id

This integer parameter is required.

The **cust_id** 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
```

profile_id

This integer parameter is required.

The **profile_id** parameter contains the **Object Reference ID** of the desired FTP Profile. The FTP Profile contains the necessary information to connect to the FTP server (server URL, username, password, etc.).

Example:

```
"profile_id": 40
```

file_path

This string parameter is optional.

The **file_path** parameter is part of the total filepath to the location on the FTP server where the import file will be stored. The complete URL and filepath to the appropriate file on the FTP server comprises three separate parameters:

- **Domain**: This portion of the total filepath is actually part of the associated FTP Profile item, and not the FTP Import Template. When you assign the FTP Profile (see **profile_id** above), the system uses the URL contained within the **domain_name** parameter in that item. For example:

```
"domain_name": "sftp://sftp.cheetah.com"
```

- **Folder Structure**: The folder structure component of the filepath is contained within the **file_path** parameter. For example:

```
"file_path": "/incoming/daily/"
```

- **File Name**: The file name is contained with the **file_mask** parameter (see **file_mask** below). The file name can also optionally contain a "mask," that's automatically appended to the file name. The best practice is to include a date / time stamp as a file mask, order to keep the file names unique. For example:

```
"file_mask": "IINFL100_DAILY_{(yyyy-MM-dd)}.csv"
```

Together, all three parameters make up the location and name of the inbound file:

```
sftp://sftp.cheetah.com/incoming/daily/IINFL100_DAILY_{(yyyy-MM-dd)}.csv
```

The value you enter in the **file_path** parameter should include just the folder structure on the server.

For example:

```
"file_path": "/incoming/daily/"
```

file_mask

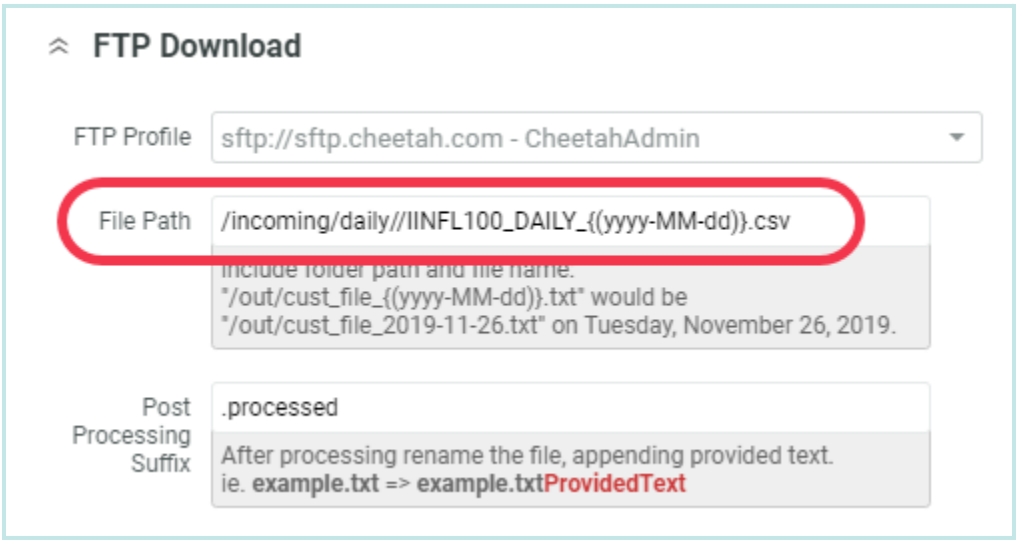
This string parameter is optional.

As described above, the **file_mask** parameter is part of the total filepath to the location on the FTP server where the import file will be stored. The value you enter in the **file_mask** parameter should include just the file name, and optionally also a file mask, if you're using one.

For example:

```
"file_mask": "IINFL100_DAILY_{(yyyy-MM-dd)}.csv"
```

When looking at an FTP Import Template in the Messaging application, the values from the **file_path** and **file_mask** parameters will be concatenated, and appear within the "File Path" text field.



processed_suffix

This string parameter is optional.

If provided, the **processed_suffix** value will be appended to the end of every inbound file name. This suffix is used to differentiate processed files from new files, and also to prevent any confusion around duplicate file names.

The file suffix can also optionally contain a "mask," that's automatically appended to the file suffix.

For example:

```
"processed_suffix": ".processed_{(yyyy-MM-dd)}"
```

delete_flag

This integer parameter is optional.

The **delete_flag** parameter is used to indicate if you want the platform to delete the file from the FTP site after it's been downloaded. Provide a value of "1" in this parameter if you

want the platform to delete the file. If you don't provide this parameter, the system defaults to a value of "0."

For example:

```
"delete_flag": 1
```

timestamp_flag

This integer parameter is optional.

The **timestamp_flag** parameter is used to indicate if you want the platform to check the file's "last modified" date and time prior to importing the file. This option is intended to prevent the platform from re-importing the same data file. Provide a value of "1" in this parameter if you want the platform to perform this check. If you don't provide this parameter, the system defaults to a value of "0."

For example:

```
"timestamp_flag": 1
```

pgp_flag

This integer parameter is optional.

The **pgp_flag** parameter is used to indicate if you're using PGP encryption on the inbound file. Provide a value of "1" in this parameter if the file is encrypted (you'll also need to provide your PGP encryption key in the **pgp_key_id** parameter; see below for details). If you don't provide this parameter, the system defaults to a value of "0."

For example:

```
"pgp_flag": 1
```

pgp_key_id

This string parameter is optional.

The **pgp_key_id** parameter is used to provide your PGP encryption key.

For example:

```
"pgp_key_id":
"mQENBFstRJQBCADVNhkfhgBSGz1KOuXMfDGrMmx03KDkA89FQcJocKoW5fTRZopIuMBsb
b+BZAxQ30Fuy4uCRtyHB/UE+43zaZ6FuaUzt38jRZx/fWAD91HnuBpC4TFwjfP/3JidlGe
xEW7HgufwEFTdM7HUWvYDCLMY0f77gJ3pqaRXmMb713j1H4BOC0Pq7Ph3Mhd7w0vDt7o5v
```

```
u9krYCayQxY6AoebTegAXUp+zKDLJZ/0pJ1TJuwO8jSk8q87EF4CvAdQVG7Ux77xWHL8We
AlKRkqLDTaoCoKLYTe1Y+UU72ZlcxUOYRAgnyKwwEV5sVjxpU+"
```

zip_seq

This integer parameter is optional.

The **zip_seq** parameter is used to indicate if you're using compression on the inbound file. Provide a value of "1" in this parameter if the file is compressed. If you don't provide this parameter, the system defaults to a value of "0."

For example:

```
"zip_seq": 1
```

zip_password

This integer parameter is optional.

The **zip_password** parameter is used if you're importing a compressed, password-protected import file. This parameter should contain the password needed to access the file.

For example:

```
"zip_password": "Ch33tah!"
```

schedule_id

This integer parameter is required unless you're providing the schedule details within the payload of the API message (see **Parameters -- Schedule** for details).

The **schedule_id** parameter represents the Schedule ID for the desired processing schedule for this FTP Import Template. For more information on how to create a reusable Schedule object, please see the *Schedule API Technical Guide*.

For example:

```
"schedule_id": 67429
```

obj

This object contains the name of the new FTP Import Template.

Example:

```
"obj":
```

```
{
  "display_name": "Test FTP Import Template API"
}
```

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 FTP Import Template. This name must be unique within your account.

Parameters -- Files to Import

The options and parameters described in this section explain how Messaging should process the import file once it's been downloaded from the FTP server. For example, this section includes what to name the file, what Data Map to use, etc.

ftpImports

This object provides the detailed file-handling options for each import file being processed by this FTP Import Template. You may reference one or more files within the **ftpImports** object since an FTP Import Template can optionally be used to import multiple files within the same ZIP file. You can choose to handle all of these files in the same manner, or you can select different configuration options and Data Maps for each file.

For example:

```
"ftpImports": [
  {
   "file_mask": "*.*",
   "parent_obj_id": 37249,
   "seq": 1,
   "import": {
      "cust_id": 394,
      "type_id": "FILE",
      "import_file": "daily_import",
      "prop_map_id": 3451,
      "hierarchy_id": 100,
      "task_priority": 300,
      "ncoa_process_id": 35,
      "importPreviewFlags": {
```

```
            "skip_parsed_preview": 1
          }
        }
      }
    ]
```

file_mask

This string parameter is required.

The **file_mask** parameter contains the name of the import file for which you're defining the specifications. This field is needed if you're importing a ZIP file containing multiple files, and you want to define different specifications for each file.

If you're not importing a ZIP file with multiple files, or if you want to treat all files in the same manner, then you can simply enter a default **file_mask** value of "*.*".

Example using the default value:

```
"file_mask": "*.*"
```

Example specifying a file within the ZIP file:

```
"file_mask": "*.inputfile1.*"
```

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 this import file. If you don't provide this parameter, the system will save the file in the default folder location for your account, which is typically the top-most Folder in your folder structure.

Please note that this parameter defines where you're saving the import file, and not where you're saving the FTP Import Template asset. FTP Import Templates are considered "non-foldered" items, meaning they're not saved in a user-specified folder location.

Example:

```
"parent_obj_id": 37249
```

seq

This integer parameter is required.

The **seq** parameter represents a counter used to define the processing sequence for which file is loaded into the database first, which file is loaded second, and so forth. If you're only importing one file, just use a value of "1" in this parameter.

```
"seq": 1
```

import

This object contains a variety of parameters that control the processing specifications for an import file. The parameters in this object are described below in more detail.

cust_id

This integer parameter is required.

The **cust_id** 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:

```
"cust_id": 394
```

type_id

This string parameter is required.

For an import file, the value in this parameter must be "FILE."

Example:

```
"type_id": "FILE"
```

import_file

This string parameter is optional.

The **import_file** value is used as a suffix to a system generated file name in the platform's back-end processing.

This value isn't displayed anywhere within the application. For audit and tracking purposes, the best practice is to populate this parameter with the original filename. In

this manner, if any questions arise about the source of the import, you can more easily trace the request message back to the original file that contained the inbound data. If you don't provide this parameter, the system uses a default value of "ftp_template."

Example:

```
"import_file": "daily_import"
```

task_priority

This integer parameter is optional.

The **task_priority** parameter allows you to establish the priority for your import. The priority you select determines the order in which your import will be handled. Generally speaking, your imports with a higher priority will take precedence over your imports with a lower priority. If your imports have the same priority, they will be processed in the order in which they were added (older imports first). The valid values for this parameter are:

- "800:" Urgent
- "700:" High
- "300:" Normal (this is the default value used if you don't provide this parameter)

Example:

```
"task_priority": 700
```

prop_map_id

This integer parameter is required.

The **prop_map_id** parameter represents the **Object Reference ID** of the Data Map that you want to use to control how, and where, the inbound data is written.

Example:

```
"prop_map_id": 4569
```

ncoa_process_id

This integer parameter is optional.

The **ncoa_process_id** parameter represents the **Object Reference ID** of an NCOA Process. NCOA Processes are used by clients running Print Campaigns, to run postal addresses through the National Change of Address system.

Example:

```
"ncoa_process_id": 21
```

hierarchy_id

This integer parameter is optional.

The **hierarchy_id** parameter is used in Parent / Child database architectures to determine how data is loaded, and what fields each "Child" view can access. The possible values are:

- "100" -- Update Current Customer Only: This option will load the data to the Parent system only.
- "200" -- Update Current Customer and Children: This option will load the data to the Parent and to any Child systems as well.

> **Note**
>
> The FTP IMPORT TEMPLATE endpoint doesn't support the ability to select a custom Hierarchy Rule.

importPreviewFlags

This object contains flags related to the import process.

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

skip_parsed_preview

This integer parameter is optional.

The **skip_parsed_preview** flag is used to enable the "Auto-commit" function. Auto-commit allows you to load the file to the database without requiring manual review and approval of the parsing results. The valid values for this parameter are:

- "1" -- Enable the Auto-commit feature for this file.

- "0" -- Disable the Auto-commit feature for this file (default value).

5 View or Edit an FTP Import Template

Overview

This section describes how to work with existing FTP Import Templates via GET, PUT, or DELETE requests to the FTP IMPORT TEMPLATE endpoint.

Retrieve an FTP Import Template

The GET method is used to retrieve information about a single FTP Import Template.

The request message must include the FTP Import Template's **Object Reference ID**. The Object Reference ID must be sent as a query type parameter within the URL.

For example:

```
https://api.eccmp.com/services2/api/FtpTemplate?id=842
```

Edit an FTP Import Template

The PUT method allows you to submit modifications to an existing FTP Import Template. Using this method, you can change any aspect of the FTP Import Template.

The request message must include the FTP Import Template's **Object Reference ID** and the desired changes. The Object Reference ID must be sent as a query type parameter within the URL.

For example:

```
https://api.eccmp.com/services2/api/FtpTemplate?id=842
```

All of the other FTP Import Template parameters are sent within the body of the message. The parameters for the PUT method are the same as described in the **Create an FTP Import Template** section, with the following additions, described below.

ftp_template_id

This integer parameter is required.

The **ftp_template_id** represents the FTP Import Template's **Object Reference ID**. The value you provide in this parameter must match the **id** value sent within the URL.

Example:

```
"ftp_template_id": 842
```

ftpTemplateAction

This string parameter is optional; this parameter is available only in a PUT request, and not in a POST request.

The **ftpTemplateAction** is used to start or stop the processing schedule for an FTP Import Template. The valid values are:

- "SAVE" -- Save the FTP Import Template.
- "START" -- Start the processing schedule for the FTP Import Template.
- "STOP" -- Stop the processing schedule for the FTP Import Template.

Delete an FTP Import Template

The DELETE method allows you to delete an existing FTP Import Template.

The request message must include the FTP Import Template's **Object Reference ID**. The Object Reference ID must be sent as a query type parameter within the URL.

For example:

```
https://api.eccmp.com/services2/api/FtpTemplate?id=842
```

6 Response

This section describes the possible response messages sent back from the FTP PROFILE and FTP IMPORT TEMPLATE endpoints.

Success

The success responses for the FTP PROFILE and FTP IMPORT TEMPLATE endpoints are described below.

FTP Profile

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

A successful response to a GET message to retrieve all FTP Profiles will generate a response code of "200," followed by a list of all FTP Profiles in your account, sorted by Object Reference ID.

A successful response to a GET message to retrieve a single FTP Profile will generate a response code of "200," followed by the details of the specified FTP Profile.

A successful response to a PUT message will generate a response code of "200," followed by the details of the modified FTP Profile 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.

FTP Import Template

A successful response to a POST message will generate a response code of "200," followed by the details of the new FTP Import Template 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 FTP Import Template.

A successful response to a PUT message will generate a response code of "200," followed by the details of the modified FTP Import Template 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 FTP PROFILE or FTP IMPORT TEMPLATE request message, the platform will send an "error" message with details of the problem. Below is a list of error codes and their descriptions.

FTP Profile Errors

Response Code	Error message	Description
400	"An Obj with this name already exists"	Duplicate FTP Profile name. The **profile_name** value must be unique within your client account.
400	"An FtpProfile with this ID does not exist."	Invalid or unknown Object Reference ID.
400	An Id is required when updating an object, Ids must match on the updated object.	Mismatch between the Object Reference ID in the URL and the value in the **profile_id** parameter.
400	The passwords do not match	In a PUT request, when updating the password, the values in **password** and **password_confirm** must match.
400	Enter the password and confirm the password	In a PUT request, you must provide both the **password** and the **password_confirm** parameters.
400	User Name is required	In a POST request, **user_name** parameter is required.
400	Profile Name is required	In a POST request, **profile_name** parameter is required.

Response Code	Error message	Description
400	Password is required	In a POST request, **password** parameter is required.
400	Domain Name is required	In a POST request, **domain_name** parameter is required.

FTP Import Template Errors

Response Code	Error message	Description
400	An Obj with this name already exists	Duplicate FTP Import Template name. The **display_name** value must be unique within your client account.
400	An Id is required when updating an object, Ids must match on the updated object.	In a PUT request, the Object Reference ID in the URL must match the value in the **ftp_template_id** parameter.
400	An FtpTemplate with this ID does not exist	In a GET request, the Object Reference ID idoes not exist.
400	This FtpTemplate has been deleted	In a GET request, the FTP Import Template was previously deleted.
400	Invalid data was provided.	FTP Template is null.
500	The current FTP Schedule will cause the Import to never download any files	The FTP Template's next scheduled run time does not exist.
500	You must include a Schedule or Schedule Id when creating an FtpTemplate	You must use either the **schedule_id** parameter (to reference an existing schedule) or the **scheduleControl** object (to define a new schedule) when creating a new FTP Import Template
500	You provided both a Schedule and a Schedule Id. You may only submit one of them.	Both a schedule and **schedule_id** are provided.
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.

Response Code	Error message	Description
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.

7 Sample Messages

This section contains several sample request and messages for the FTP PROFILE and FTP IMPORT TEMPLATE endpoints.

Sample Request #1

This POST request creates a new FTP Profile.

JSON Payload

```
{
  "profile_name": "Test FTP Profile API",
  "user_name": "CheetahAdmin",
  "password": "qaCh33tah!",
  "password_confirm": "qaCh33tah!",
  "domain_name": "sftp://sftp.cheetah.com",
  "passive_mode_flag": 1
}
```

Sample Request #2

This POST request creates a new FTP Import Template with one import file.

```
{
    "cust_id": 394,
    "profile_id": 45,
    "file_path": "/inbound/daily",
    "file_mask": "import_file_{(yyyy-MM-dd)}.csv",
    "processed_suffix": ".done_{(yyyy-MM-dd)}",
    "delete_flag": 0,
    "timestamp_flag": 0,
    "pgp_flag": 0,
    "zip_seq": 1,
    "scheduleControl": {
        "startTime": "2018-09-20T11:16:00",
        "endTime": "2018-10-08T00:00:00",
        "timeZone": "Central_Standard_Time",
        "dayFrequency": {
            "frequencyType": "Daily",
            "daysInterval": 4
        },
        "timeFrequency": {
```

```
            "timeIntervalType": "MultipleTimesADay",
            "multipleTimesInterval": {
               "runIntervalUnit": "Hour",
               "runInterval": 3
            }
        }
    },
    "ftpImports": [
        {
            "file_mask": "*.*",
            "parent_obj_id": 37249,
            "seq": 1,
            "import": {
                "cust_id": 394,
                "type_id": "FILE",

                                        "import_file": "daily_import",
                "task_priority": 300,
                "prop_map_id": 2592,
                "hierarchy_id": 100,
                "importPreviewFlags": {
                    "skip_parsed_preview": 1
                }
            }
        }
    ],
    "obj": {
        "display_name": "Test API FTP Import Template"
    }
}
```

8 Appendix A -- Identifiers

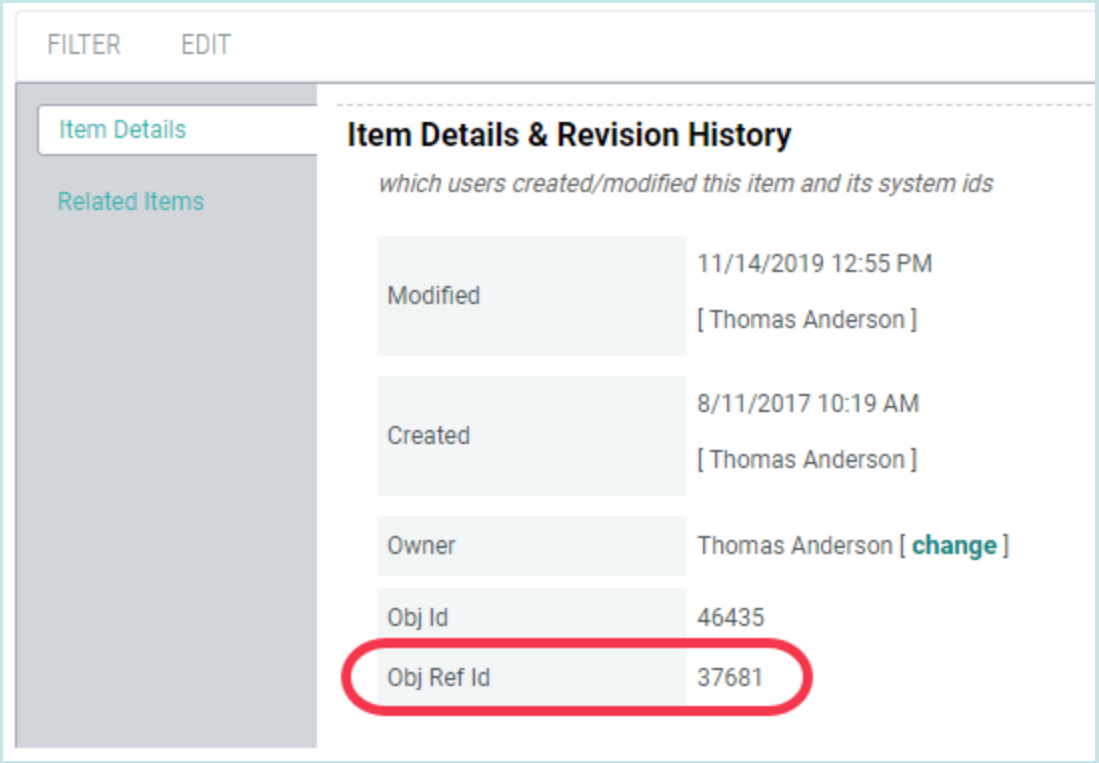
Messaging uses several different types of IDs when referencing assets, such as tables, fields, folders, Filters, and so forth. This appendix describes these different types of IDs, and provides steps on how to look up the value of an ID.

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"
}
```

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"
}
```

9 Appendix B -- Parameter Values

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

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
Venezuela_Standard_Time	West_Asia_Standard_Time
Paraguay_Standard_Time	India_Standard_Time

timeZone	timeZone
Atlantic_Standard_Time	Sri_Lanka_Standard_Time
Central_Brazilian_Standard_Time	Nepal_Standard_Time
SA_Western_Standard_Time	Central_Asia_Standard_Time
Pacific_SA_Standard_Time	Bangladesh_Standard_Time
Newfoundland_Standard_Time	N_Central_Asia_Standard_Time
E_South_America_Standard_Time	Myanmar_Standard_Time
Argentina_Standard_Time	SE_Asia_Standard_Time
SA_Eastern_Standard_Time	Nth_Asia_Standard_Time
Greenland_Standard_Time	China_Standard_Time
Montevideo_Standard_Time	Nth_Asia_East_Standard_Time
UTC_02	Singape_Standard_Time
Mid_Atlantic_Standard_Time	W_Australia_Standard_Time
Azes_Standard_Time	Taipei_Standard_Time
Cape_Verde_Standard_Time	Ulaanbaatar_Standard_Time
Mocco_Standard_Time	Tokyo_Standard_Time
UTC	Kea_Standard_Time
GMT_Standard_	Yakutsk_Standard_Time
Greenwich_Standard_Time	Cen_Australia_Standard_Time
W_Europe_Standard_Time	AUS_Central_Standard_Time
Central_Europe_Standard_Time	E_Australia_Standard_Time
Romance_Standard_Time	AUS_Eastern_Standard_Time
Central_European_Standard_Time	West_Pacific_Standard_Time
W_Central_Africa_Standard_Time	Tasmania_Standard_Time
Namibia_Standard_Time	Vladivostok_Standard_Time
Jdan_Standard_Time	Central_Pacific_Standard_Time
GTB_Standard_Time	New_Zealand_Standard_Time
Middle_East_Standard_Time	UTC_12
Egypt_Standard_Time	Fiji_Standard_Time
Syria_Standard_Time	Kamchatka_Standard_Time

timeZone	timeZone
South_Africa_Standard_Time	Tonga_Standard_Time