MARIGOLD™

API Technical Guide: HTTP Post

Cheetah Messaging

Table of Contents

1 Introduction

Purpose

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

Overview

The HTTP POST feature is used to submit data to your Messaging database via an HTTP POST message. Once received by the platform, this message can also optionally be used to trigger the deployment of an Event-triggered Campaign.

This feature does not support authentication, nor does it support JSON or XML. Unlike most of the other API endpoints in Messaging, the data is not sent in the payload of a JSON or XML message. Instead, the data is sent as name / value pairs within a single query string using URL-encoded values.

The request can optionally be secured by using the secure HTTPS protocol. If using HTTPS, please note that the connection is encrypted, but the data being passed is not encrypted.

The URLs for this feature are:

- **North America:** https://ats.eccmp.com/ats/post.aspx?cr=[xxx]&fm=[xxx]&[data fields]
- **Europe:** https://ats.eccmp.eu/ats/post.aspx?cr=[xxx]&fm=[xxx]&[data fields]
- **Japan:** https://ats.eccmp.jp/ats/post.aspx?cr=[xxx]&fm=[xxx]&[data fields]

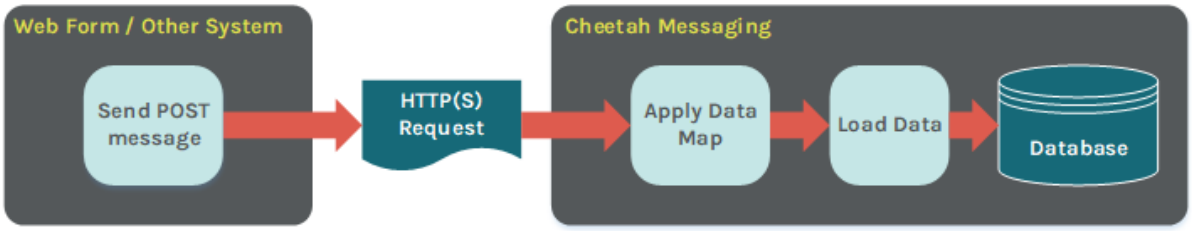
Use Cases

The HTTP POST feature supports two main use cases:

- **Recipient-submitted data**: In this use case, you have a Web Form used to collect information from your customers. When the customer submits the Form, the data is sent to Messaging via the HTTP POST feature.
- **System-to-system data**: In this use case, there's no user-facing Web Form. Instead, this use case is intended as a system-to-system method for sending data to Messaging. The platform uses an API Post to provide a unique URL for the submission, along with the description of the expected data within the message.

From a technical perspective, both of the above uses cases are very similar. The only real difference is who's providing the data. If you need a user-friendly front-end for your customers to enter information, you'll need to create a Web Form. If you're sending data from some other system to Messaging, you can use the API Post method.

The following diagram depicts the basic processing flow for loading data via HTTP POST.

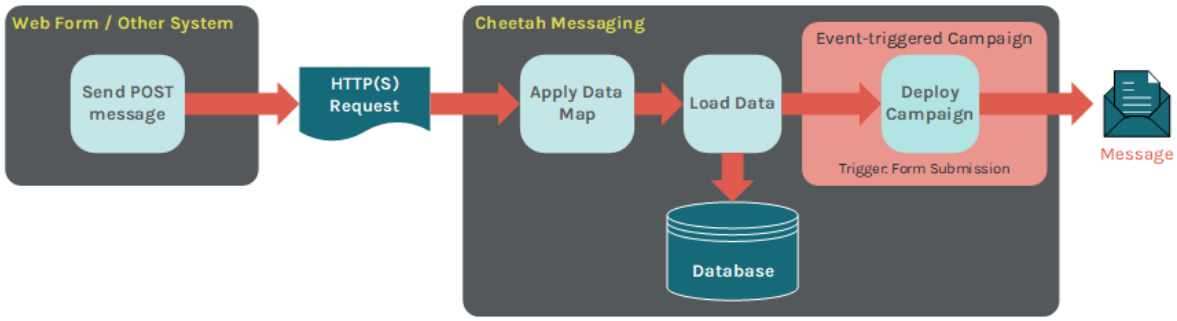


Campaign Trigger

An HTTP POST message can optionally be used as a triggering mechanism for the deployment of an Event-triggered Campaign in Messaging. The trigger type for the Campaign must either be a Web Form or an API Post. The Campaign is deployed after the data in the request message is successfully written to the database, which allows you to use data from your database when building your message content.

When this endpoint is utilized in this fashion, it's typically referred to as the HTTP POST EVENT.

The following diagram depicts the basic processing flow for loading data and deploying a Campaign via an HTTP POST message.



Pre-requisites

The HTTP POST feature requires that the following assets be defined within your Messaging account:

- **Data Map** -- The Data Map provides data handling instructions, such as where to store the inbound data contained within the request message, whether this data should be used to update existing records and / or create new records, and any optional formatting or special processing to perform on the inbound data.
- **API Post** -- If data is being submitted from another system, then you must create and publish an API Post within Messaging. The API Post defines all the expected fields in the request message, and provides support for custom response messages and a schedule. Please note that the API Post must have the "REST API Only" option and the "Triggered" option both unchecked.
- **Web Form** -- If recipients are submitting data via a Web Form, then you must create and publish the Web Form. Web Forms provide a user-friendly, front-end interface for your customers to enter their information. The Web Form can optionally be hosted within the Messaging platform, or within your system, or on some other third-party system.

If you're using HTTP POST to trigger the deployment of an Event-triggered Campaign, you must have the following asset defined as well:

- **Campaign** -- You must define and launch an Event-triggered Campaign that uses either the above API Post, or the above Web Form, as the trigger type.

2 Request

Overview

This section describes how to send data to Messaging using the HTTP POST feature.

Parameters

Parameters are the required or optional information contained within the HTTP request message, and tell the system what data to write to the database. All these parameters are sent as name / value pairs within the URL. For example:

```
https://ats.eccmp.com/ats/post.aspx?cr=394&fm=2678&s_email=johndoe@cheetahdigital.com&s_name_first=John&s_name_last=Doe
```

These parameters are described below in more detail.

cr

This integer parameter is required.

The **cr** 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 (many 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:

```
cr=394
```

fm

This integer parameter is required.

The **fm** parameter represents the **Form ID** of the API Post or Web Form.

Example:

```
fm=2678
```

cn

This integer parameter is optional.

The **cn** parameter represents the **Running Campaign ID** of a Campaign that you want to relate to the record in the post. This parameter is typically used for reporting purposes, so you can attribute the record in the post back to a specific Campaign.

In most cases, you don't need to provide this parameter, and the best practice is to use the platform's "Tracking Parameters" feature instead. Tracking Parameters are a set of built-in, system-generated parameters that will automatically pass the Campaign information for you, so you don't need to manually add the **cn** parameter to the post.

If you are using the **cn** parameter, please note that this ID is different from the Campaign's Object Reference ID, which is typically used in other Messaging endpoints.

Example:

```
cn=28595
```

mg

This integer parameter is optional.

The **mg** parameter represents the "Message ID" of the email message that you want to relate to the record in the post. The Message ID is a system-generated identifier for every message deployed from Messaging. This parameter is typically used for reporting purposes, so you can attribute the record in the post back to a specific message.

As with the **cn** parameter above, the **mg** parameter is typically not needed, and best practice is to use Tracking Parameters instead.

Example:

```
mg=4567890
```

ri

This integer parameter is optional.

The **ri** parameter represents the "Record ID" (also known as the Primary Key ID, or PKID) of the message recipient. The Record ID is a system-generated identifier for every record in Messaging. This parameter is typically used for reporting purposes, so you can attribute the record in the post back to a specific individual recipient.

As with the **cn** parameter above, the **ri** parameter is typically not needed, because the post should include the recipient's Unique Identifier field, or fields (also know as the Alternate Key ID, or AKID).

Example:

```
ri=345678
```

Data Fields

Each field being sent in the HTTP POST message (i.e., email address, first name, last name, etc.) must be submitted as name / value pairs in the URL, using the following rules:

- For the field names, you must use the system-generated **Column Name**, and not the viewer-friendly "Display Name."
- Single-value fields must use a prefix of “s_” followed by the Column Name.
- Multi-value fields must use a prefix of "m_” followed by the Column Name.
- Depending on the client configuration, the Unique Identifier field(s) must be submitted within the HTTP POST, so that the platform can correctly identify if the data is an existing record to be modified, or a new record to be created.

Example:

```
s_email=johndoe@cheetahdigital.com
```

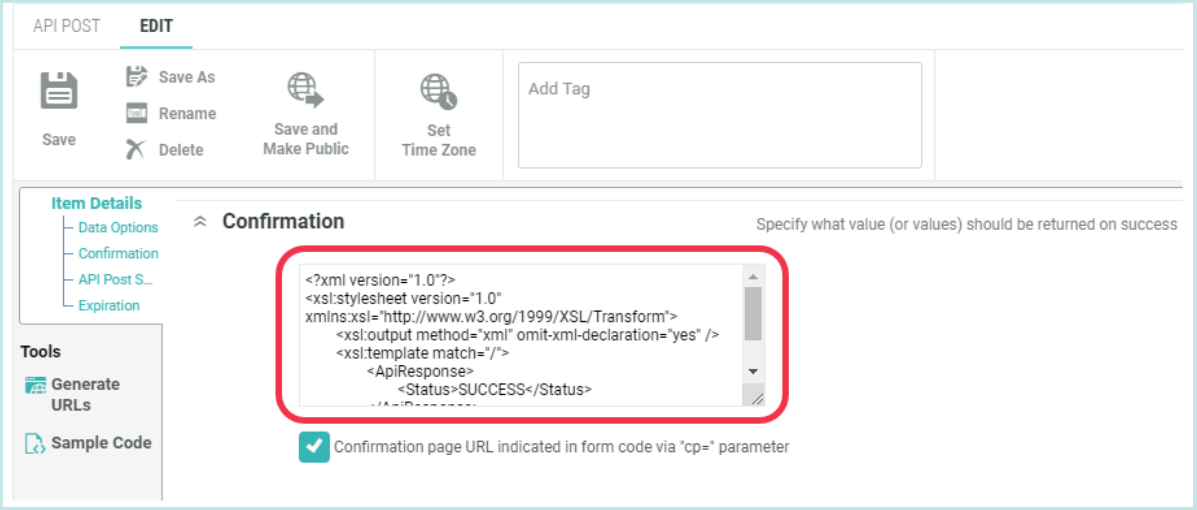
3 Response

This section describes the possible response messages sent back from an HTTP POST message.

Success

Upon successful completion of an HTTP POST request, a "success" message is returned with a response code of '200."

Optionally, if you're using an API Post, you can define a custom response message. On the API Post screen, enter the XML code for your custom response message in the "Confirmation" text field.



Errors

If Messaging encounters a problem with an HTTP POST request message, the platform will reply with a standard HTTP error code.

4 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.

Form ID

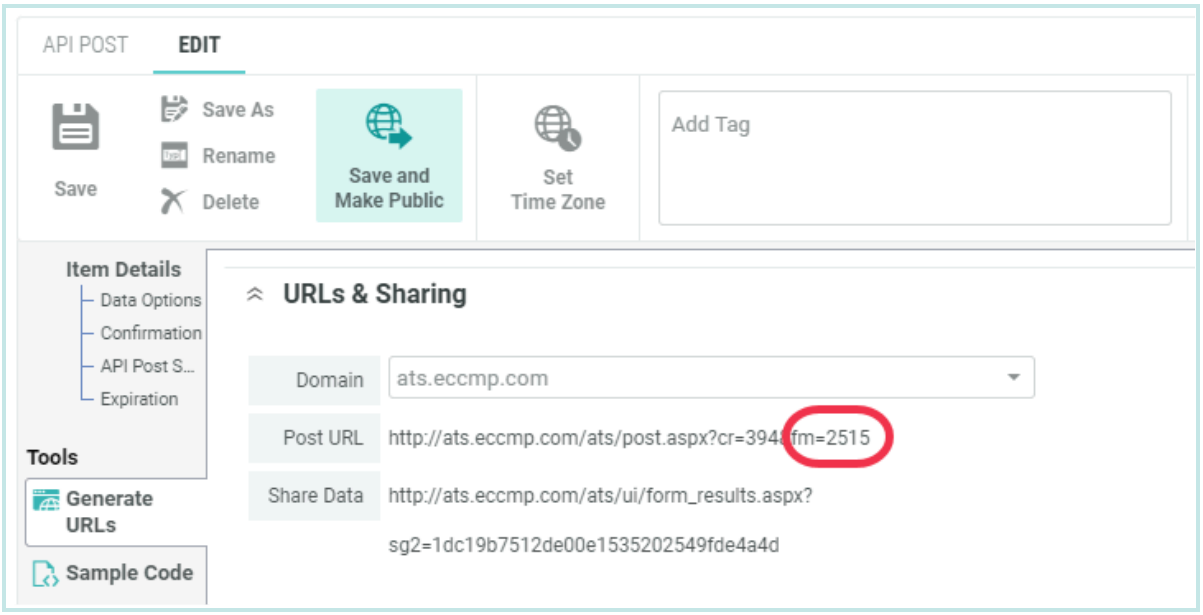
The Form ID is a system-generated identifier for every API Post and Web Form in your account.

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

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

To find the Form ID within the application:

1. From the System Tray, select *Data Integration* > *Processes* > *API* Posts (for API Posts) or *Web* > *Management* > *Web Forms* (for Web Forms). The system displays a list of all the assets of that type in your account.
2. Select the desired asset. The asset details screen is displayed.
3. From the Function Menu, select "Generate URLs." Within the "Post URL" field or "Form URL" field, the system displays the Form ID as the value of the "fm" parameter.



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

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 name or its type. For example, to retrieve information about all of your API Posts, use the asset type "ImportFormApi."

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

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 Form ID, which is referred to as the "**ref_id**." For example:

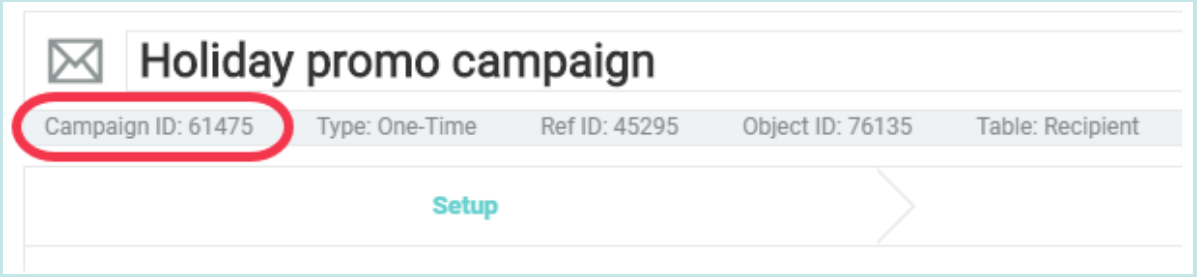
```
{
  "obj_id": 44737,
  "display_name": "Test API Post",
  "type_id": "ImportFormApi",
  "ref_id": 2515,
  "parent_obj_id": 37249,
  "eligibility_status_id": "READY"
}
```

Running Campaign ID

The "Running Campaign ID" is a system-generated identifier for a Campaign, that gets created only when the Campaign is launched.

The value for the Running Campaign ID can be found on the Campaign details screen within the Messaging application:

1. From the System Tray, select *Campaigns* > *Management* > *Campaigns*.
2. Browse to and select the desired launched Campaign. The Campaign details screen is displayed.
3. The Running Campaign ID is displayed near the top of the screen.



Optionally, you can use the CAMPAIGN STAT endpoint, which will return the Running Campaign ID in the response message:

1. Submit a GET request to the CAMPAIGN STAT API endpoint. The request must include the Campaign's Object Reference ID. For example:

```
https://api.eccmp.com/services2/api/CampaignStat?campId=31896
```

2. As part of the API response message, the system provides the Running Campaign ID, which is referred to as the "**campId**." For example:


```
{
    "campId": 31897,
    "mergeSetupTime": "2018-06-15T15:53:59.13",
    "dmsSetupTime": "2018-06-15T15:53:59.137",
    "rtsSetupTime": "2018-06-15T15:53:59.143",
    "inbSetupTime": "2018-06-15T15:53:59.243",
    "msgListCreationStatusId": 700,
    "msgListCreationStartTime": "2018-06-15T15:54:01.237",
    "msgListCreationFinishTime": "2018-06-15T16:07:58.323",
    "msgCreatedAmount": 3,
    "contCalculationStatusId": 700,
```

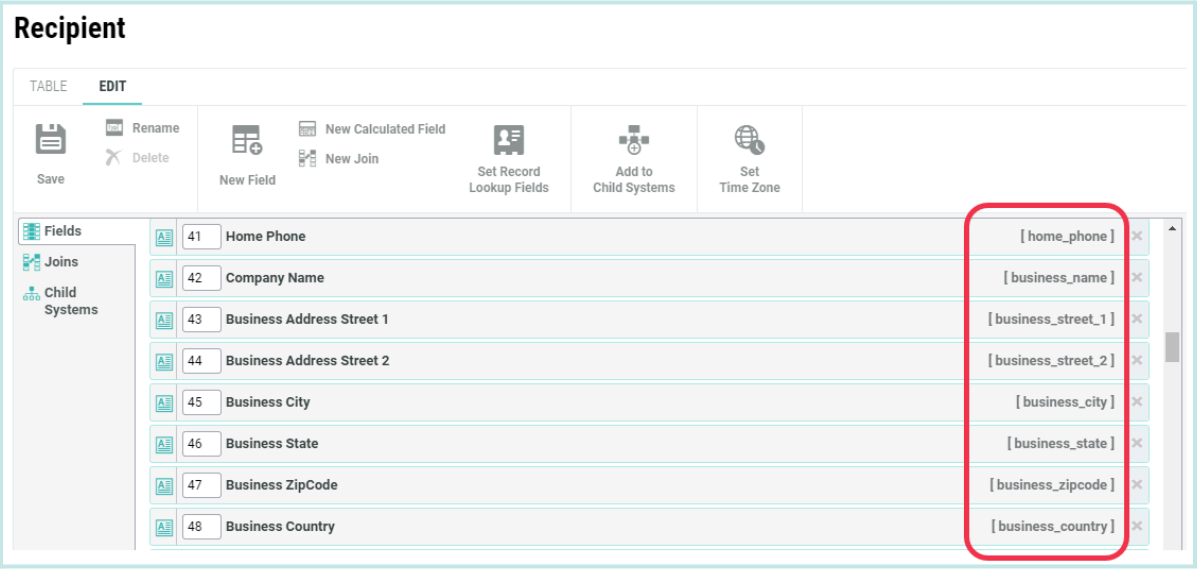
```
    "contCalculationStartTime": "2018-06-15T15:54:53.58",
    "contCalculationFinishTime": "2018-06-15T16:07:58.323",
    "contCalculatedAmount": 3,
    "personalizationStatusId": 700,
    "personalizationStartTime": "2018-06-15T15:54:53.58",
    "personalizationFinishTime": "2018-06-15T16:07:58.323",
    "personalizedAmount": 3,
    "sendingStatusId": 700,
    "sendingStartTime": "2018-06-15T15:54:53.58",
    "sendingFinishTime": "2018-06-15T16:07:58.323",
    "sentAmount": 3
}
```


Column Name

The Column Names for fields can be found on the Tables screen within the Messaging application, or by using the TABLE endpoint.

To look up the Column Name within the Messaging application:

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



Optionally, to retrieve the Column Name for a field using the TABLE endpoint:

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

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

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

Sample Response:

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

Note

When adding the Column Name to your HTTP POST, you must prefix the Column Name with either "s_" (for single-value fields) or "m_" (for multi-value fields).