Webhook Capabilities

Cheetah Digital

Last Modified: November 2021

Version History

Version	Date	Description	Reviewed / Approved by
1.0	March 2018	Initial release	Cheetah Digital Product Management
1.1	November 2018	Minor updates	Cheetah Digital Product Management
1.2	March 2019	Minor updates	Cheetah Digital Product Management
1.3	June 2019	Minor updates	Cheetah Digital Product Management
1.4	July 2019	Minor updates	Cheetah Digital Product Management
1.5	September 2019	Minor updates	Cheetah Digital Product Management
1.6	November 2019	Minor updates	Cheetah Digital Product Management
1.7	November 2021	Review, added version history	Cheetah Digital Product Management

Table of Contents

1 Introduction

Overview

The purpose of this document is to provide a high-level overview of the Webhook capabilities offered by the Cheetah Messaging platform.

A Webhook is a communication mechanism that allows a web application to send information to another system. The information is transmitted via an HTTP Request similar to submitting a web form from a browser.

Webhooks and API Calls

Webhooks are conceptually similar to API calls in that they both provide a means of passing data between two systems. Like API calls, a Webhook is composed of four main components:

1. Endpoint – the destination system's URL
2. Headers – defines the message content type, and optionally also the message authorization
3. Method type –defines the type of operation and the format of the data being transferred
4. Data – the message content sent as parameters or in a payload

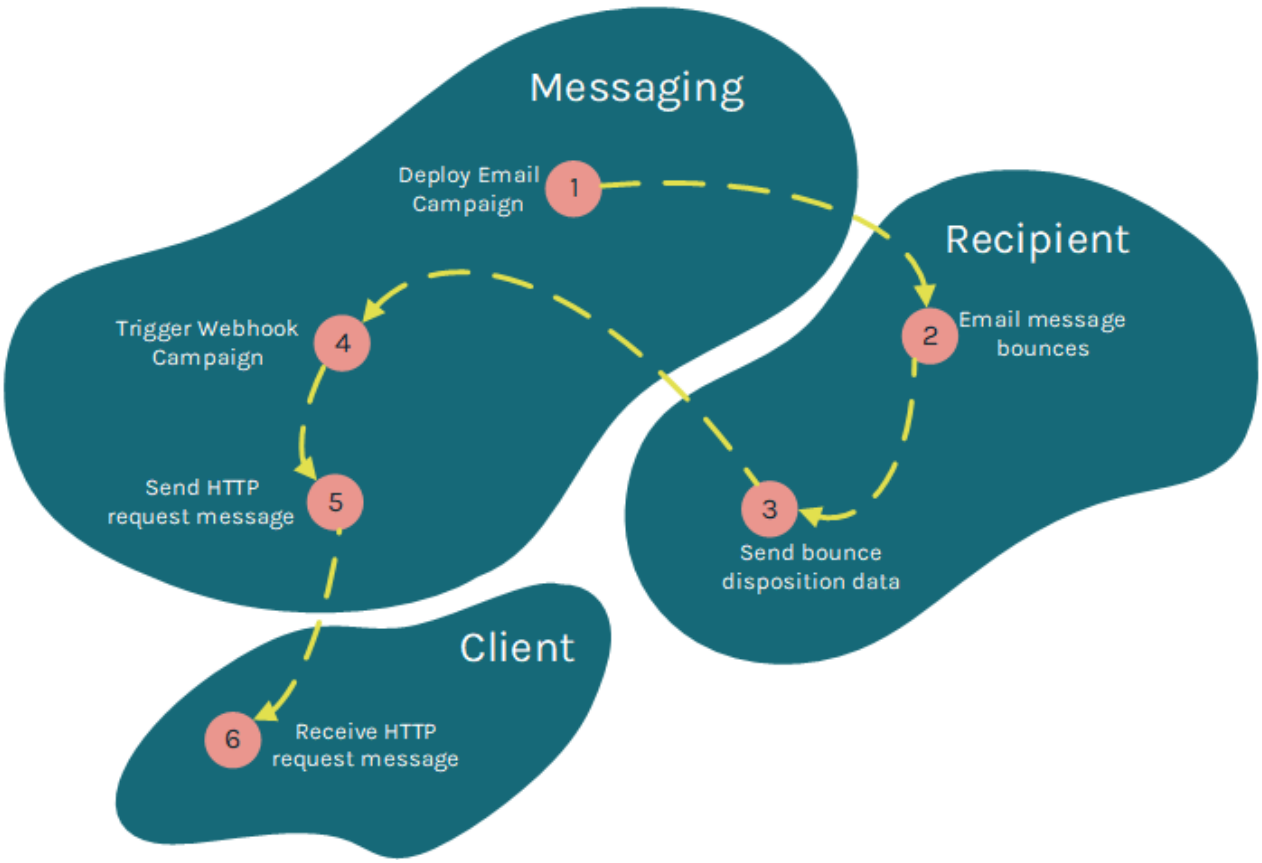
The combination of these four components offer flexible ways to connect to a wide variety of external databases and applications.

2 Process Flow

Many clients who utilize the Webhook feature will deploy a Webhook message based on some consumer action, or triggering event. For example, you could update your internal database based on consumer email “read” or “click” activity. Or you could update a sales system with new contact information from a new subscriber. When the triggering event is identified, the platform will deploy the Webhook Campaign, and send an HTTP request containing information about the event to your internal system, or to some other third-party.

As an example, let's say you send out order confirmation email messages to your consumers. In the event that one of these messages bounces, you want to be notified so you can follow up with the consumer through another channel, such as over the phone or by SMS text. You could set up a Webhook Campaign that uses "Email Bounce" as its trigger type.

When the system identifies a bounced email, it would trigger the deployment of the Webhook Campaign to send an HTTP request message to your internal customer service system. This message could be configured to include the consumer's name, order information, and phone number. The following diagram depicts this process flow.



3 Messaging Set-Up

Overview

This section describes the various options available when configuring the Webhooks feature in Messaging.

Campaigns

Within Messaging, Webhooks are treated as a Campaign channel. Similar to other channels, Webhook Campaigns can be run on-demand, scheduled, or triggered based on some specified event. You define Filters which determine the audience based on business rules or logical criteria.

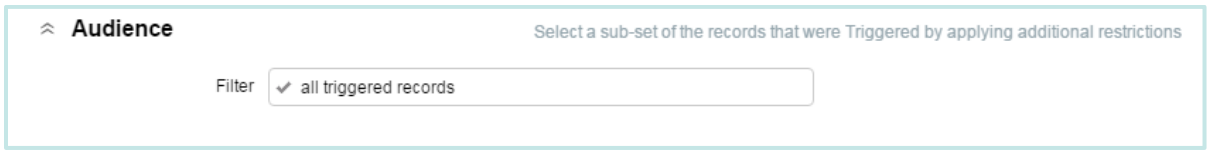
While Webhooks can be used to make batch requests, they are typically triggered by an event to send an update message to a client or third-party database. Messaging supports a wide range of trigger types, such as email opens, clicks, or bounces, web form submissions, opt-out requests, and mobile keyword texts, for example.

Filters

Your Webhook Campaign includes a Filter, which establishes the business rules for identifying the desired audience or user records. The audience represents which record, or records, will be used to populate the HTTP request payload.

If your Webhook Campaign is an Event-triggered Campaign, then you can simply use the default Filter option, which is “All triggered records.” Whatever record triggered the

deployment of the Webhook Campaign will be included within the HTTP request message. Or you can define a custom Filter with other business logic.



Request Endpoint

The destination endpoint is entered into the “Request URL” field. For example, let’s say the destination URL for your HTTP request message is:

```
http://sample.com/post.aspx
```

You would enter this URL in the “Request URL” field on the Campaign screen, as shown here.

Please note that request messages can optionally be secured by using the HTTPS protocol.

HTTP Method

The platform supports the following HTTP methods when sending messages through the Webhook channel:

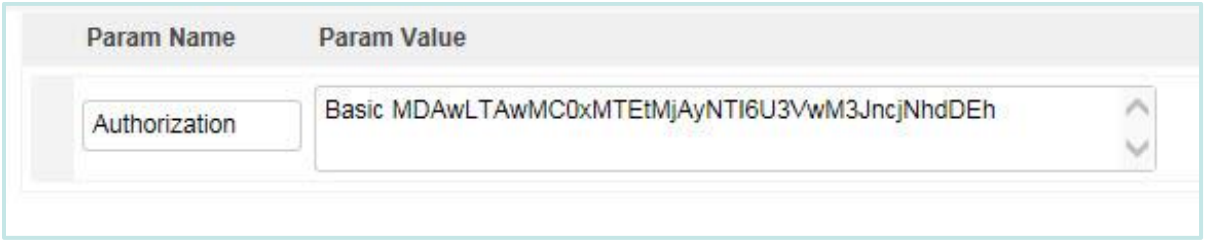
- GET: Used to read or retrieve data without directly modifying it.

- POST: Used to create a new asset.
- PUT: Used to update an asset (if it already exists), or create an asset (if it doesn't already exist).
- PATCH: Used to partially modify an existing asset.

- HEAD: Similar to GET, used to read or retrieve data, but without the response body. Instead, only the headers are returned.
- DELETE: Used to delete an existing asset.

HTTP Headers

Header key / value pairs are determined by the destination platform. They are commonly used to specify the message content type and to authorize the sender of the message. As an example, you can define an Authorization key, and include the encoded username / password as the value, as shown here.



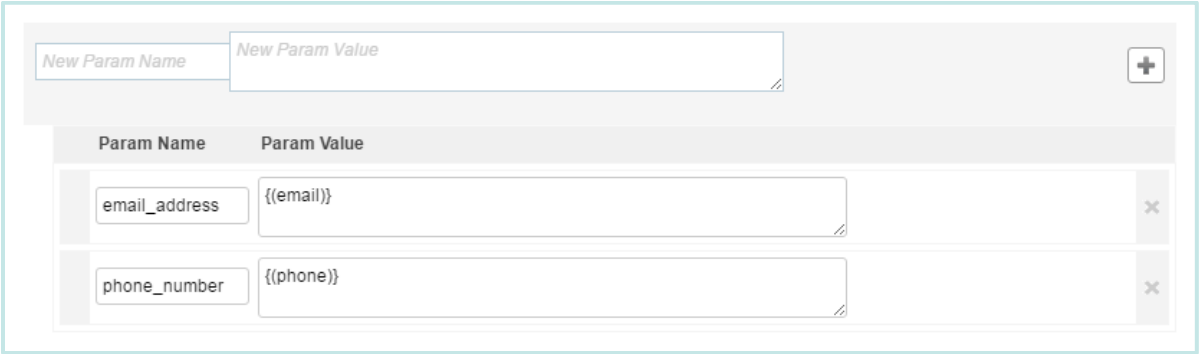
Data Type

Messaging supports two “Data Type” options that control the format of the HTTP request message you want to send. The options (described below) are Key Value Pair and Body Text.

Key Value Pair

The Webhook Campaign screen allows you to build a simple, custom HTTP request message containing whatever data fields are needed. The fields and their associated values are submitted as key-value pairs, contained within the header of the HTTP request, rather than in the body.

As an example, let's say you created an Event-triggered Webhooks Campaign that uses "Email Bounce" as the trigger type. When the platform identifies a bounced email, the Webhooks Campaign will deploy, and submit that customer's email address and phone number to your customer service system.



The selected parameters (i.e., the fields in the destination database) are included in the message, as shown in the following example. The blue text indicates the endpoint URL, the green text indicates the parameters, and red text indicates the values being passed in the message.

```
http://sample.com/post.aspx?email_address={(email)}&phone_number={(phone})
```

By using Merge Symbols like {(email)} as the parameter value, the system will pass the value in that database field within the message.

Conversely, if you "hard-code" a specific value in the message, the system will always pass that value in every message.

```
http://sample.com/post.aspx?email_address=jsmith@company.com&phone_number={(phone})
```

In most cases, you want to use Merge Symbols, in order to dynamically populate the Webhook message with information pulled from the specified field in your Messaging database.

Body Text

The Body Text message option allows you to define a text payload, usually JSON, that gets sent in the body of the message, rather than in the header. This option allows you to send more structured data.

If using the Body Text option, enter your JSON payload in the “Body” text field.