Webhook Instructions

Developer Guide

Last Updated 8/20/26

Summary

This document provides a quick tutorial on how to integrate with webhooks using the CSP API. The example shown in this document assumes that the developer will subscribe to webhook events in the Brand category.

CSPs can see a full list of webhook events, their descriptions, triggers, and more in the TCR System Events documentation.

Changes in This Document Update:

- Updated with the latest HMAC information. Also reformatted the document based on current styling.

Table of Contents

Summary **1**
Table of Contents **2**
Document History **2**
Process Flow **3**
Create an API Key 3
Authenticate on the CSP API 4
Subscribe to Notifications 4
Confirm Webhook Subscriptions 5
Testing 5
Delete a Webhook Subscription 6
How to Validate a Webhook Signature Using HMAC **7**

Document History

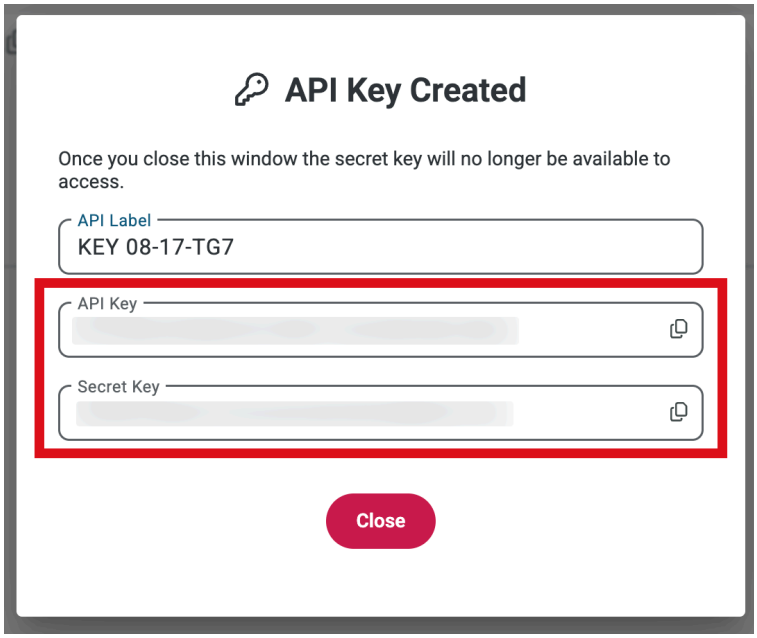
Date	Comment	Author
8/20/2026	Updated with the latest HMAC information. Also reformatted the document based on current styling.	Victor Cardoso

Process Flow

The following screens show you how to use the CSP API Swagger implementation for subscribing, testing, and removing subscriptions. As a developer, you can invoke the endpoints directly to perform these actions.

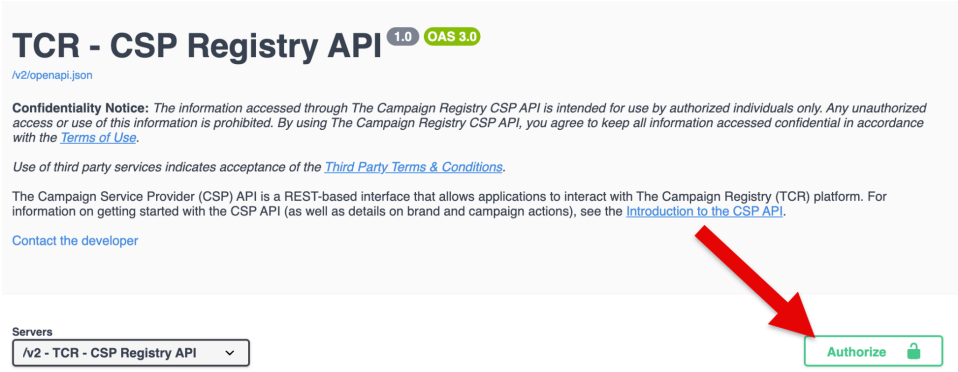
Create an API Key

1. Sign in to the CSP Portal.
2. In the sidebar, click **Integrations**.
3. Click **Create API Key** to create a new API key.
4. Make note of the **API Key** and **Secret Key** fields. You will not be able to view the secret key after it has been created.



Authenticate on the CSP API

1. Navigate to the CSP Swagger API for the environment you are targeting. For the Staging environment, this would be: https://csp-api-staging.campaignregistry.com/v2/restAPI
2. Click **Authorize** and login with your API credentials, using the API Key as the username and the Secret Key as the password.



Subscribe to Notifications

1. Navigate to the **`PUT /webhook/subscription`** endpoint.
2. Click **Try It Out**.
3. Change the request body to contain the event category to subscribe to and the webhook endpoint to receive the notifications. For example:

```
{
      eventCategory: "BRAND"
      webhookEndpoint: "https://webhook.site/###"
}
```

4. Click **Execute**.
5. Repeat these steps for every event category of interest.

Confirm Webhook Subscriptions

1. Navigate to the **`GET /webhook/subscription`** endpoint.
2. Click **Try It Out**.
3. Click **Execute**.
4. The response should contain a list of your webhook subscriptions.

Testing

1. Navigate to the **`POST /brand/nonBlocking`** endpoint and click **Try it Out**.
2. change the request body to contain some test brand information. For example:

```
{
  "entityType": "PRIVATE_PROFIT",
  "firstName": "John",
  "lastName": "Doe",
  "displayName": "ABC Mobile",
  "companyName": "ABC Inc.",
  "ein": "111111111",
  "einIssuingCountry": "US",
  "phone": "+12024567890",
  "street": "123 6th Ave",
  "city": "New York",
  "state": "NY",
  "postalCode": "10001",
  "country": "US",
  "email": "johndoe@abc.com",
  "stockSymbol": "ABC",
  "stockExchange": "NASDAQ",
  "ipAddress": "string",
  "website": "http://www.abcmobile.com",
  "brandRelationship": "BASIC_ACCOUNT",
  "vertical": "RETAIL",
  "altBusinessId": "string",
```

```
  "altBusinessIdType": "string",
  "referenceId": "string",
  "mock": false,
  "tag": [],
  "mobilePhone": "+12024567890",
  "businessContactEmail": "johndoe@abc.com"
}
```

3. Click **Execute**.
4. Now check your webhook endpoint to see if it received the BRAND_ADD event.

Delete a Webhook Subscription

Note that this will only remove the selected webhook subscriptions associated with the API key used for authorization.

1. Navigate to the **`DELETE /webhook/subscription/{eventCategory}`** endpoint.
2. Click **Try It Out**.
3. Select the event category to unsubscribe from:
4. Click **Execute**.

How to Validate a Webhook Signature Using HMAC

To establish the authenticity of webhook events, TCR uses the Hash-based Message Authentication Code (HMAC). HMAC allows authentication without revealing the secret key used by both parties. This ensures data integrity and that the message is not tampered with. The following formula is how we generate the "X-Registry-Signature-256" authentication header:

Signature=Base64(HMAC_SHA256(secret_key, webhook_endpoint+canonicalized_json))

Please note that the webhook event JSON needs to be canonicalized based on the RFC8785 standard. In cryptography, operations like hashing require data to be expressed in an invariant format. A single character or order difference will result in completely different hash values.

The message is composed of both "webhook_endpoint" and "canonicalized_json" so that users can verify the event is intended for the targeted webhook endpoint and the event payload is not tampered with.

The following is an example in Java on how to generate the signature:

Code

```
//This library consists of HMAC tools
<dependency>
     <groupId>commons-codec</groupId>
     <artifactId>commons-codec</artifactId>
     <version>1.15</version>
</dependency>
// JSON Canonicalization is used to create a single predictable
// byte stream
<dependency>
     <groupId>io.github.erdtman</groupId>
     <artifactId>java-json-canonicalization</artifactId>
     <version>1.1</version>
</dependency>
```

```
package org.example;
import org.apache.commons.codec.digest.HmacAlgorithms;
import org.apache.commons.codec.digest.HmacUtils;
import org.erdtman.jcs.JsonCanonicalizer;
import javax.xml.bind.DatatypeConverter;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
public class HmacDemo {
     public static void main(String[] args) throws Exception {
         String apiKeySecret = "mysecret";
         String webhookEndpoint =
         "https://webhook.site/4838d9df-6ee2-4dde-ae75-87527c51f7e
         ";
         // JSON file consists of JSON retrieved from event
         String json = new
         String(Files.readAllBytes(Paths.get("/tmp/test.json")),
         StandardCharsets.UTF_8);
         JsonCanonicalizer jsonCanonicalizer = new
         JsonCanonicalizer(json);
         byte[] hmac = new HmacUtils(HmacAlgorithms.HMAC_SHA_256,
         apiKeySecret)
         .hmac(webhookEndpoint +
         jsonCanonicalizer.getEncodedString());
         String signature =
         DatatypeConverter.printBase64Binary(hmac);
         System.out.println(json);
         System.out.println(signature);
     }
}
```

Event Payload and SHA256 Signature


```
{
  "cspId" : "SPQXYIG",
  "brandName" : "zz",
  "campaignId" : "C8PWPWB",
  "brandReferenceId" : "adfert111",
  "brandId" : "B5HDZSA",
  "description" : "Campaign C8PWPWB for brand B5HDZSA (zz) is provisioned in the registry",
  "mock" : false,
  "eventType" : "CAMPAIGN_ADD",
  "cspName" : "tcr",
  "campaignReferenceId" : null,
  "channelId" : "10DLC"
}
iEhlJC6m3bBxepsij8yt/VJAJK6PJ9nwAsehXfiqrRk=
```


Webhook Verification