BETTER MESSAGING
campaign registry

Introduction to the CSP API

Developer Guide

Last Updated 8/7/25

Table of Contents

Document History

Date	Comment	Author
3/18/2025	Initial publication.	Victor Cardoso
4/3/2025	Clarified wording on declined campaigns.	Victor Cardoso
4/8/2025	Minor corrections.	Victor Cardoso
4/15/2025	Corrected some hyperlinks that were pointing to the wrong source.	Victor Cardoso
4/28/2025	Updated the Deleting a Brand section to indicate that deleted brands can now be found via the `GET /brand/{brandId}` endpoint.	Victor Cardoso
5/21/2025	Removed reference to mock brands and campaigns being automatically deleted after 30 days. This feature is not yet implemented.	Victor Cardoso
8/7/2025	Changed language for Auth+ 2.0 and added link to feature overview.	Victor Cardoso

Overview

The Campaign Service Provider (CSP) API is a REST-based interface that allows applications to interact with The Campaign Registry (TCR) platform. It's intended for use by CSPs with prior development experience to automate functions related to brand and campaign management.

The CSP API works by sending and receiving data as JavaScript Object Notation (JSON) objects and uses standard HTTP response codes, authentication, and verbs. In addition to the live Production environment, TCR offers a Staging environment to customers for testing, which doesn't affect live data or incur charges associated with brands, campaigns, or interactions with third parties.

Key Concepts

Data Model

The TCR data model contains three key entities:

1. **Campaign Service Provider.** The CSP is a business or organization who is contracted with one or more Direct Connect Aggregators (DCAs) and TCR. The CSP is typically an entity who hosts campaigns on their own technology platform and uses TCR's tools to register those campaigns for delivery on mobile networks. To qualify as a CSP, organizations must first register with TCR and undergo a vetting process that includes reference checks.
2. **Brand.** A brand is a legal entity responsible for the messaging content in a campaign. A brand relies on their CSP to submit campaign requests to TCR. All brands registered by a CSP undergo an identity verification process to ensure they are a legitimate entity who qualifies for specific messaging use cases and terms with mobile network operators (MNOs).
3. **Campaign.** A campaign is a package of information that includes details about the CSP, brand, and an intended use case for messages sent across mobile networks. Sample messages, links, keywords, and opt-in/opt-out information are also included, depending on the use case. Campaigns can only be submitted by CSPs.

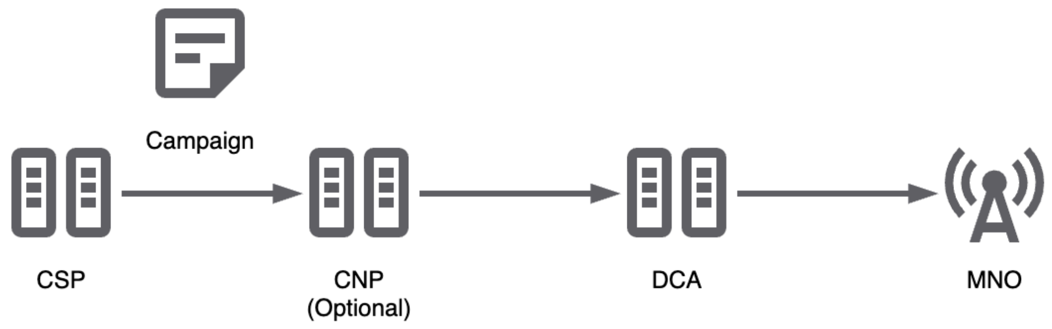
✏️ **Note:** *TCR also supports the concept of a Reseller, an entity that sits between the brand and the CSP. Resellers do not have a direct relationship with TCR and are specified as an attribute during the campaign registration process. This helps provide a complete picture of the organizations involved in the campaign's creation and delivery.*

Connectivity Chain

The connectivity chain is the path a campaign takes from the brand (who creates the messaging campaign) to the mobile network operator (who eventually distributes the messages to end subscribers). It represents the different Connection Network Providers (CNPs) and aggregators that are part of a campaign's journey. CNP is an umbrella term that can refer to either CSPs or DCAs.

This is explained in more detail under the Campaign Operations section.

Figure 1. *Connectivity Chain*



Unique Identifiers

CSP, Brand, and Campaign objects are each assigned a unique seven-character, uppercase alphanumeric identifier in the TCR platform:

- CSP identifiers begin with the letter S (e.g., S123ABC).
- Brand identifiers begin with the letter B (e.g., B123ABC).
- Campaign identifiers begin with the letter C (e.g., C123ABC).

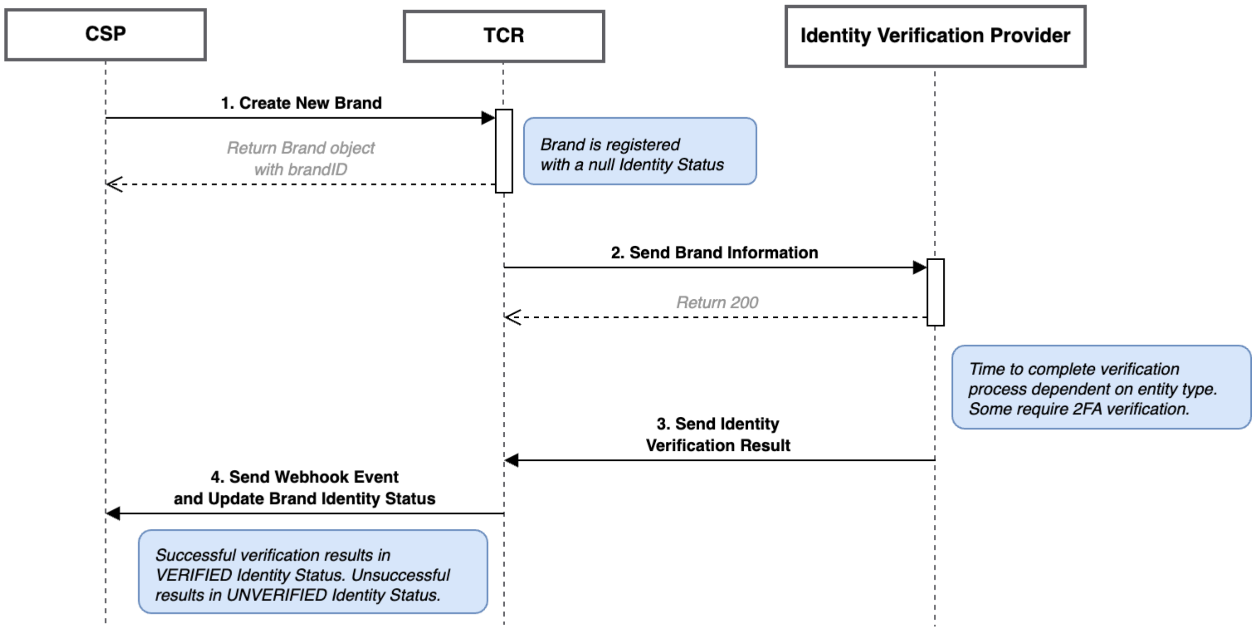
Brand Operations

Registering a New Brand

A CSP must first register a brand before they can start creating campaigns. When a brand is registered via the CSP API (using the `POST /brand/nonBlocking` endpoint), they undergo a verification process with a TCR-approved identity vetting provider. When the identity vetting provider returns a result, the CSP is notified that a brand is ready for campaign registration via a BRAND_IDENTITY_STATUS_UPDATE webhook event (see the Webhooks and Event Notifications section for more information). The typical response time for this initial identity verification check is under 2 minutes, but can take longer depending on the operational state and workload of the identity vetting provider or if the entity type of the brand requires further verification (as is the case for Public Profit brands and Sole Proprietorships). Only after a successful identity verification process is a brand allowed to create campaigns.

The following diagram shows a visual representation of a typical brand registration process.

Figure 2. *New Brand Registration*


CSP
TCR
Identity Verification Provider
1. Create New Brand
Return Brand object with brandID
Brand is registered with a null Identity Status
2. Send Brand Information
Return 200
Time to complete verification process dependent on entity type. Some require 2FA verification.
3. Send Identity Verification Result
4. Send Webhook Event and Update Brand Identity Status
Successful verification results in VERIFIED Identity Status. Unsuccessful results in UNVERIFIED Identity Status.

Identity Verification Results

TCR's identity vetting provider will attempt to verify the identity of a brand using the information submitted by the CSP. Based on the identity provider's response, the brand will be assigned a VERIFIED or UNVERIFIED identity status. VERIFIED brands are eligible to begin registering campaigns immediately, although the types of available campaigns might be limited based on their brand attributes (see the section Brand Optional Attributes for more information). If a brand receives an UNVERIFIED identity status, then the identity provider was unable to verify their identity or the brand's identity type requires further verification, and the CSP must take additional action. Brands with an UNVERIFIED identity status are not allowed to register any campaigns on the TCR platform.

✏️ **Note:** *Sole Proprietorships are special types of entities that undergo a different verification process. For more information, see Working with Sole Proprietorships. Public Profit entities are also subject to additional identity verification. For more information on Public Profit identity verification, see Authentication+.*

Potential Brand Identity Statuses

The following identity statuses are possible for a given brand:

- **UNVERIFIED:** TCR's identity vetting provider was unable to verify or identify the existence of the brand. An Unverified brand is not permitted to register new messaging campaigns. **Note:** *Sole Proprietorships and Public Profit brands may also receive an Unverified status if they do not complete their additional identity verification processes.*

- **VERIFIED:** TCR's identity vetting provider was able to verify the submitted information and identify the existence of the brand. A Verified brand is permitted to run standard messaging campaigns across all MNO networks. Additional use cases or improved messaging terms are available if the brand undergoes external vetting. For more information, see External Vetting.

- **VETTED_VERIFIED:** This status is only shown for brands that have successfully applied for external vetting. A Vetted_Verified brand may gain access to higher throughput and additional use cases, depending on the MNO. For more information, see External Vetting. **Note:** *Sole Proprietor brands cannot apply for external vetting, and are therefore not eligible for a Vetted_Verified status.*

If a brand receives an Unverified status, a brand feedback mechanism is available to provide more details as to the cause (see the `GET /brand/feedback/{brandId}` endpoint for more information). CSPs can then take action to address the problem and get the brand changed to a Verified status. The following table provides a summary of possible brand feedback issues by entity type and ways to resolve them.

Entity Type	Potential Unverified Causes	Possible Recourse
PRIVATE_PROFIT	• Invalid `ein`, or a mismatch between the `ein` and `companyName` attribute.	• Appeal the brand's identity status. • Revet the brand with corrections and supporting documentation. • Apply for external vetting.
PUBLIC_PROFIT	• Invalid `ein`, or a mismatch between the `ein` and `companyName` attribute. • The brand business contact email address failed validation. • The brand business contact did not complete Authentication+ verification within 30 days of registration. • Brand attributes were updated for an existing Public Profit brand and Authentication+ verification has not yet been completed.	• Appeal the brand's identity status. • Wait for the brand business contact to complete Authentication+ verification. • Revet the brand with any corrections and supporting documentation. A revet will also trigger a new Authentication+ 2FA email.
NON_PROFIT	• Invalid `ein`, or a mismatch between the `ein` and `companyName` attribute. • The brand is not a US-based non-profit organization or does not have a US EIN.	• Appeal the brand's identity status. • Revet the brand with corrections and supporting documentation. • Apply for external vetting. • If the brand is not a US-based non-profit, apply as a Private Profit entity.

Entity Type	Potential Unverified Causes	Possible Recourse
GOVERNMENT	• Invalid ein, or a mismatch between the ein and companyName attribute. • The brand is not a US-based government organization.	• Appeal the brand's identity status. • Revet the brand with corrections and supporting documentation. • Apply for external vetting. • If the brand is not a US-based organization, apply as a Private Profit entity.
SOLE_PROPRIETOR	• One-time password (OTP) verification has not yet been completed. • The brand is not based in the US or Canada.	• Verify the sent OTP with the `PUT /brand/{brandId}/smsOtp` endpoint. • Trigger a new OTP with the `POST /brand/{brandId}/smsOtp endpoint.`

Brand Optional Attributes

Optional attributes are a collection of non-standard properties that are assigned to some brands during identity verification based on their entity types (e.g., Tax Exempt status for a 501(c)(3) Non-Profit brand). These attributes can also be applied as a result of external vetting. Optional attributes provide more information about a brand and can unlock additional use cases or different messaging terms with carriers. Optional attributes appear as an `optionalAttributes` property of the Brand object. Examples include:

- **russell3000**: Indicates a Public Profit brand that is in the Russell 3000 index. The brand may be granted higher throughput on some MNO networks.
- **taxExemptStatus**: Identifies a tax-exempt 501(c)(3/4/5/6) status for a Non-Profit brand. A 501c3 status is necessary for accessing the Charity use case.
- politicalCommitteeLocale: Identifies a type of political organization. This attribute can only be assigned by a Political vet performed by Campaign Verify. Possible values are `federal`, `state`, `local`, and `tribal`.
- **section527**: Identifies a special type of non-profit. This attribute can only be assigned by an external vet performed by Campaign Verify. A 527 group is a type of U.S.

tax-exempt organization organized under Section 527 of the U.S. Internal Revenue Code. The brand may be granted higher throughput on some MNO networks.

- **governmentEntity**: Indicates that the brand is closely affiliated, generally by government ownership or control, with federal, state, or local governments. The brand may be granted higher throughput on some MNO networks.

The list of available optional attributes can be retrieved via the `GET /enum/optionalAttributeNames` endpoint.

Authentication+

Authentication+ is an additional layer of identity verification for Public Profit brands. The primary goal is to prevent brand impersonation, which leads to potential consumer fraud such as disinformation, smishing, and spoofing.

In Auth+ 2.0, TCR further refined this process so that Auth+ verifications are unbundled from the basic identity check. Public Profit brands can now request Auth+ verifications separately from the identity status and can appeal the results. Also, additional webhook events allow CSPs to monitor the status of Auth+ verifications, providing more visibility into the workflow.

For complete information on Authentication+, including the process flow for new vs existing Public Profit brands, and applicable endpoints, see the Authentication+ 2.0 documentation.

Identity Status Appeals and Revets

As mentioned in the Identity Verification Results section, if a brand's identity status comes back as Unverified, or if the brand is missing some essential optional attributes, CSPs can appeal or revet the brand to try and resolve the problem.

1. **Performing an identity status appeal.** Appealing a brand's identity status allows the CSP to add supporting documentation to prove that the brand is a legitimate business entity or should be granted a special optional attribute (such as a Tax Exempt status or Government entity flag). CSPs can use the `POST /brand/{brandId}/appeal` endpoint to submit an appeal, but will need to provide an appeal category and should attach documentation to support their claim. For more information, see the Brand Vetting Appeals documentation.

2. **Revetting a brand.** Revetting a brand submits it again for an identity check. This should only be performed if there was missing or incorrect information supplied during brand registration (such as an incorrect company name or EIN) or if key brand information was updated after registration, resulting in the brand changing to an Unverified status. The CSP should update the brand's information and submit it for revet using the **`PUT /brand/{brandId}/revet`** endpoint. If the CSP believes the supplied information is correct, but still received an Unverified status, they should perform an identity status appeal.

External Vetting

In some instances, brands need access to special use cases (such as Political or Charity) or want to perform a deeper identity check to potentially get better messaging terms with network carriers. To accomplish this, they need to be vetted by a TCR-approved external vetting provider such as Aegis Mobile, Campaign Verify, or WMC Global. TCR can import existing vetting tokens from a TCR-approved external vetting provider, or accept a new vetting request placed via an API call.

The three types of external vets include:

- **Standard**: Standard vets can be used to verify a brand or potentially gain access to better throughput levels with network carriers. A score is returned which grants a brand access to particular levels of messaging throughput.
- **Enhanced**: Enhanced vets are typically requested if a user is dissatisfied with their Standard vet score.
- **Political**: Political vets allow brands to use the Political special use case. Non-Profit brands that have a 501(c)(3/4/5/6) Tax Exempt Status are already granted access to the Political use case and are not allowed to request or import a Political vet from any vendor.

✏️ **Note:** *Not all external vetting providers support the different types of vets. Additionally, Campaign Verify vets can only be imported.*

For detailed information on applying for external vets, see the External Vetting Requests using the CSP API documentation.

Ordering a New External Vet

1. Find the `brandId` for the brand you wish to vet.
2. Find the `evpId` for the desired vetting partner by using the `GET /enum/extVettingProvider` endpoint.
3. Find the desired `vettingClass` for the type of vet to request using the `GET /enum/vettingClass` endpoint.
4. Use the `POST /brand/{brandId}/externalVetting` endpoint to initiate an external vet. Refer to the endpoint for any required information. Political vets require an additional payload.

While this API call will return a message, the actual results of an external vet may take up to 48 hours depending on the type of vet and the vetting provider. When the vetting is complete, the results are recorded in TCR. The brand's Vetting Status may change to UNSCORE, ACTIVE, or FAILED depending on the result. The status of external vets can be polled by the CSP using the `GET /brand/{brandId}/externalVetting` endpoint.

Importing an Existing Vet

1. Find the `brandId` for the brand whose vet you want to import.
2. Find the `evpId` for the vetting partner by using the `GET /enum/extVettingProvider` endpoint.
3. Locate the `vettingId` supplied by the vetting partner.
4. Use the `PUT /brand/{brandId}/externalVetting` endpoint to import the vetting record.

This API call will return a response with the status of the latest vetting record.

Deleting a Brand

If a brand needs to be deleted from TCR, CSPs can use the `DELETE /brand/{brandId}` endpoint.

Warning: Deleting a brand is permanent and cannot be reversed. A deleted brand cannot be reactivated.

In order to delete a brand, all campaigns associated with that brand must first be deactivated by using the **`DELETE /campaign/{campaignId}`** endpoint. Deleted brands will be returned by the **`GET /brand`** or **`GET /brand/{brandId}`** endpoints.

Expired campaigns associated with deleted brands also remain searchable using the **`GET /campaign`** endpoint.

Working with Sole Proprietorships

Brands without a business EIN can be registered in TCR using the SOLE_PROPRIETOR entity type. This entity type and use case are limited to small businesses and individuals without a business EIN and who have limited traffic needs. Sole Proprietors are subject to the following limitations:

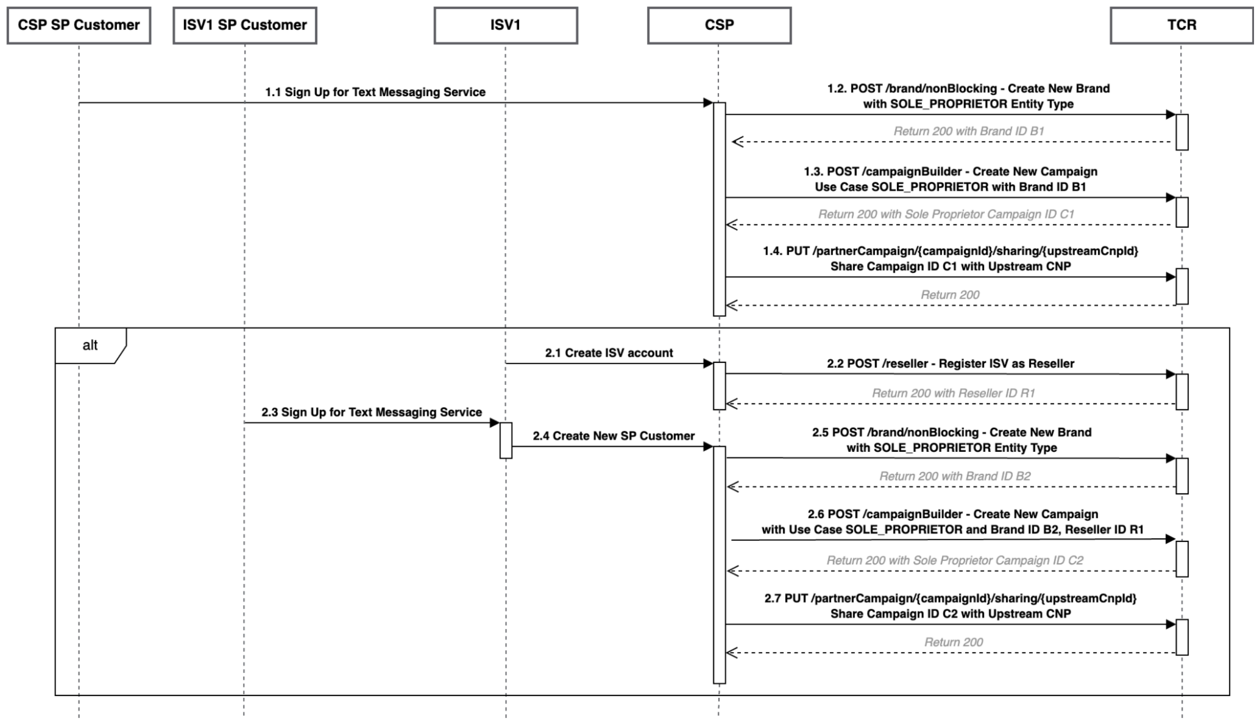
- A lower message throughput. Sole Proprietors are limited to 1,000 msg/day (T-Mobile) and 15 msg/minute (AT&T) per campaign. CSPs must enforce these daily limits.
- Can only register a single campaign using the SOLE_PROPRIETOR use case. This use case supports mixed messaging content.
- Limited to one associated telephone number (TN).
- Is not eligible to apply for external vetting to gain a higher messaging throughput.

Additionally, if the Sole Proprietor campaign originates from an Independent Software Vendor (ISV), the ISV must be selected as the Reseller during campaign registration.

To register Sole Proprietor brands and campaigns, CSPs must sign an additional contract by contacting support@campaignregistry.com. For additional information, please read the Sole Proprietor Policy documentation.

CSPs who are approved are required to send a monthly report on their Sole Proprietor traffic by the 7th of the following month. Reports can be sent via email to support@campaignregistry.com or by using an AWS S3 bucket. For more information, please read the Uploading Files to S3 for CSP Users documentation.

Figure 3. *Sole Proprietorship workflow*


CSP SP Customer
ISV1 SP Customer
ISV1
CSP
TCR
1.1 Sign Up for Text Messaging Service
1.2. POST /brand/nonBlocking - Create New Brand with SOLE_PROPRIETOR Entity Type
Return 200 with Brand ID B1
1.3. POST /campaignBuilder - Create New Campaign Use Case SOLE_PROPRIETOR with Brand ID B1
Return 200 with Sole Proprietor Campaign ID C1
1.4. PUT /partnerCampaign/{campaignId}/sharing/{upstreamCnpId} Share Campaign ID C1 with Upstream CNP
Return 200
alt
2.1 Create ISV account
2.2 POST /reseller - Register ISV as Reseller
Return 200 with Reseller ID R1
2.3 Sign Up for Text Messaging Service
2.4 Create New SP Customer
2.5 POST /brand/nonBlocking - Create New Brand with SOLE_PROPRIETOR Entity Type
Return 200 with Brand ID B2
2.6 POST /campaignBuilder - Create New Campaign with Use Case SOLE_PROPRIETOR and Brand ID B2, Reseller ID R1
Return 200 with Sole Proprietor Campaign ID C2
2.7 PUT /partnerCampaign/{campaignId}/sharing/{upstreamCnpId} Share Campaign ID C2 with Upstream CNP
Return 200

Campaign Operations

The Campaign Lifecycle

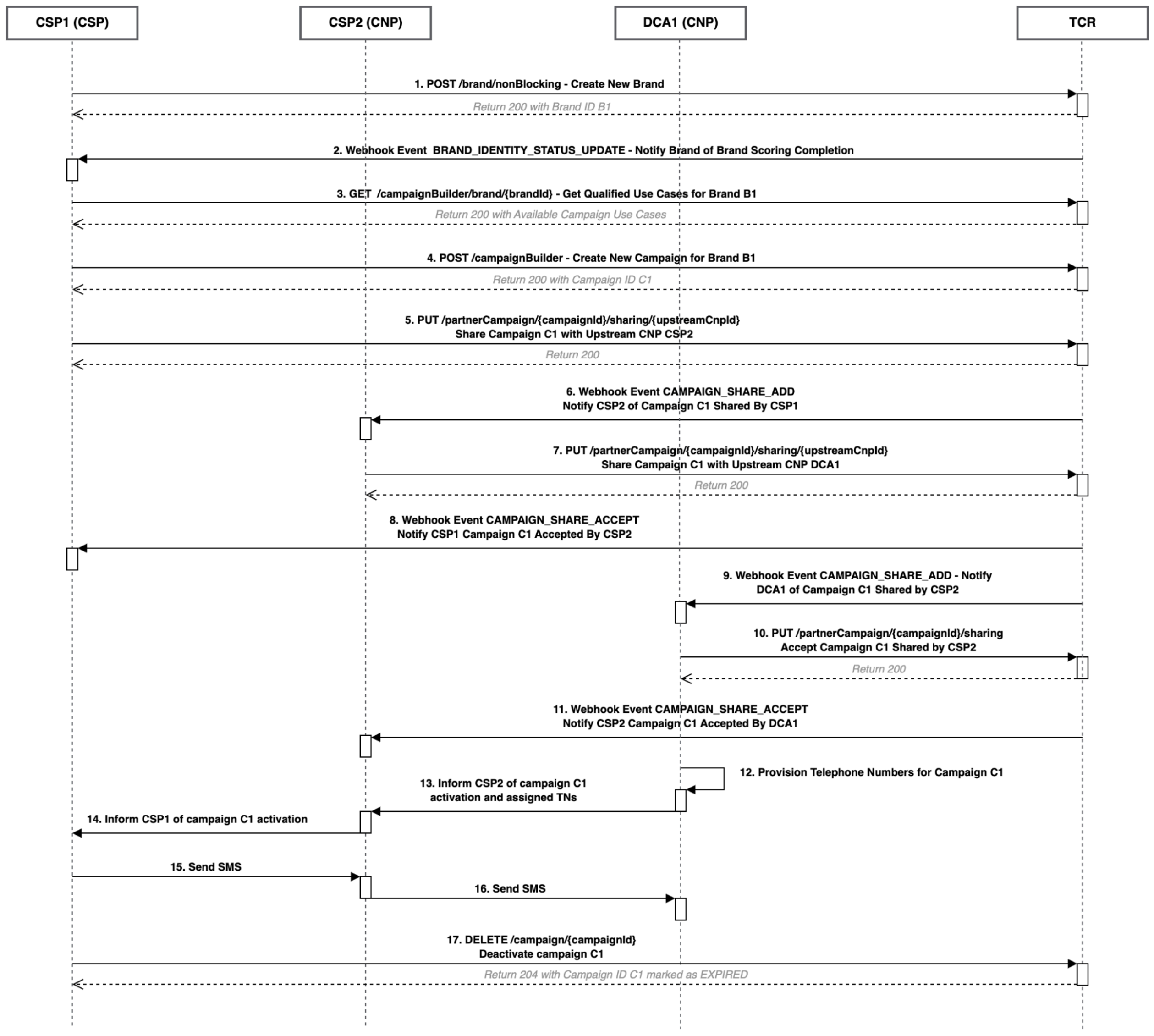
After a brand is registered and passes an initial identity verification, the CSP can begin creating campaigns. The following steps represent the different stages of a campaign's lifecycle:

1. **Campaign Registration:** CSPs register new campaigns using the Campaign Builder endpoints. Most registered brands will qualify for a majority of standard campaign use cases. However, some special use cases require external brand vetting and/or approval by the Mobile Network Operator (MNO).

2. **Campaign Sharing:** CSPs use the Partner Campaign endpoints to share a campaign with an upstream Connection Network Partner (CNP) for MNO onboarding. The CNP then assigns a phone number to the campaign using the netnumber Services Registry (nnSR) platform.

3. **Campaign Deactivation:** Active campaigns incur a monthly charge which is paid on a quarterly basis. CSPs can update campaigns to cancel auto-renewal or deactivate them to terminate the campaign immediately.

Figure 4. *Campaign Lifecycle*


CSP1 (CSP)
CSP2 (CNP)
DCA1 (CNP)
TCR
1. POST /brand/nonBlocking - Create New Brand
Return 200 with Brand ID B1
2. Webhook Event BRAND_IDENTITY_STATUS_UPDATE - Notify Brand of Brand Scoring Completion
3. GET /campaignBuilder/brand/{brandId} - Get Qualified Use Cases for Brand B1
Return 200 with Available Campaign Use Cases
4. POST /campaignBuilder - Create New Campaign for Brand B1
Return 200 with Campaign ID C1
5. PUT /partnerCampaign/{campaignId}/sharing/{upstreamCnpId}
Share Campaign C1 with Upstream CNP CSP2
Return 200
6. Webhook Event CAMPAIGN_SHARE_ADD
Notify CSP2 of Campaign C1 Shared By CSP1
7. PUT /partnerCampaign/{campaignId}/sharing/{upstreamCnpId}
Share Campaign C1 with Upstream CNP DCA1
Return 200
8. Webhook Event CAMPAIGN_SHARE_ACCEPT
Notify CSP1 Campaign C1 Accepted By CSP2
9. Webhook Event CAMPAIGN_SHARE_ADD - Notify
DCA1 of Campaign C1 Shared by CSP2
10. PUT /partnerCampaign/{campaignId}/sharing
Accept Campaign C1 Shared by CSP2
Return 200
11. Webhook Event CAMPAIGN_SHARE_ACCEPT
Notify CSP2 Campaign C1 Accepted By DCA1
12. Provision Telephone Numbers for Campaign C1
13. Inform CSP2 of campaign C1
activation and assigned TNs
14. Inform CSP1 of campaign C1 activation
15. Send SMS
16. Send SMS
17. DELETE /campaign/{campaignId}
Deactivate campaign C1
Return 204 with Campaign ID C1 marked as EXPIRED

Campaign Registration

To register a new campaign, a CSP will use the endpoints listed under the Campaign Builder section. The first step is to understand the use cases that a brand qualifies for. Each campaign must be registered with a specific use case, and some use cases require brands to have specific optional attributes, such as a confirmed Tax Exempt Status or a Government Entity flag.

To determine available use cases, requirements, and pricing, CSPs can use one of two endpoints:

- `GET /campaignBuilder/brand/{brandId}`: This returns a list of all use cases available to a specified brand.
- `GET /campaignBuilder/brand/{brandId}/usecase/{usecase}`: This endpoint will confirm whether the specified brand is qualified for a given use case. CSPs can get a list of all available use cases in TCR by using the `GET /enum/usecase` endpoint.

After determining the desired use case, the CSP should use the `POST /campaignBuilder` endpoint to register a new campaign. This endpoint contains a payload with properties that describe different elements of a campaign, some of which are required for a given use case.

✏️ **Note:** *If MMS sample files or supporting documents need to be included as part of a campaign, first register the campaign to get its Campaign ID, then use the* `POST /campaign/{campaignId}/mms` *and* `POST /campaign/{campaignId}/supportingDocument` *endpoints to attach the necessary files.*

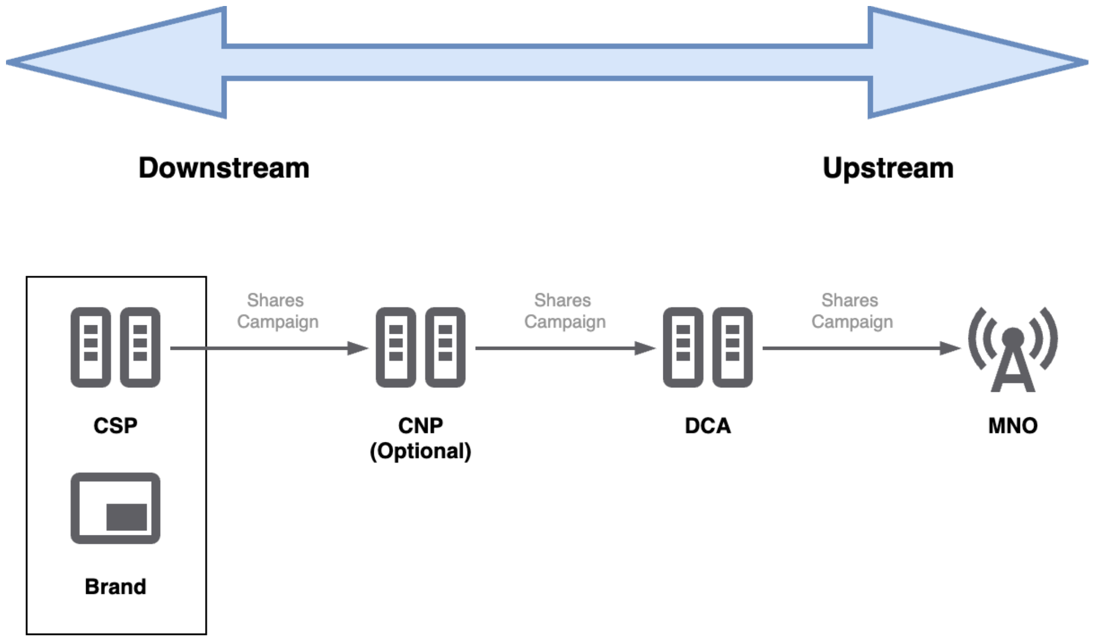
Campaign Sharing

After registering a campaign (and adding supporting files), CSPs then share the campaign with an upstream Connection Network Provider (CNP) who works with them to provision the campaign. A CNP may be another CSP registered with TCR or a Direct Connect Aggregator (DCA). The upstream CNPs in the connectivity chain will either accept or decline campaigns that are shared with them. If a campaign is declined, a reason will be provided through TCR's feedback mechanism. Once a DCA accepts a campaign, the campaign is then clear to run on the MNO network(s).

CSPs can track a campaign's approval status across upstream CNPs and MNOs selected for the campaign. Webhooks can push notifications, or CSPs can query one of the following endpoints:

- `GET /campaign/{campaignId}/sharing`: Gets the status of the campaign from any upstream CNPs.
- `GET /campaign/{campaignId}/operationStatus`: Gets the status of the campaign from MNOs selected for delivery.

Figure 5. *Connectivity Chain showing downstream and upstream sharing*



Campaign Sharing Principles

1. The term Downstream Connectivity Partner or Downstream CNP describes a CNP who has shared a campaign with you. Conversely, the term Upstream CNP describes a CNP you have shared a campaign with.
2. If a CSP has a relationship with a DCA, they can share campaigns directly to the DCA.
3. Once a campaign is shared with an upstream CNP, its sharing status changes to PENDING. While a campaign's sharing status is PENDING, the downstream CNP can't rescind the sharing request or change the upstream CNP.
4. An upstream CNP can accept or decline a sharing request from a downstream CNP. Downstream CNPs are notified of any changes to a campaign's sharing status via a webhook event.
   - Declined campaigns will show a DECLINED status. At this point, the downstream CNP can share the campaign with a different upstream CNP or the CSP can make changes to the campaign and try to appeal the rejection with the upstream CNP.
   - Accepted campaigns will change to an ACCEPTED status. This occurs when an upstream CNP either shares the campaign to another CNP, or if the upstream CNP is a DCA who accepts the campaign.

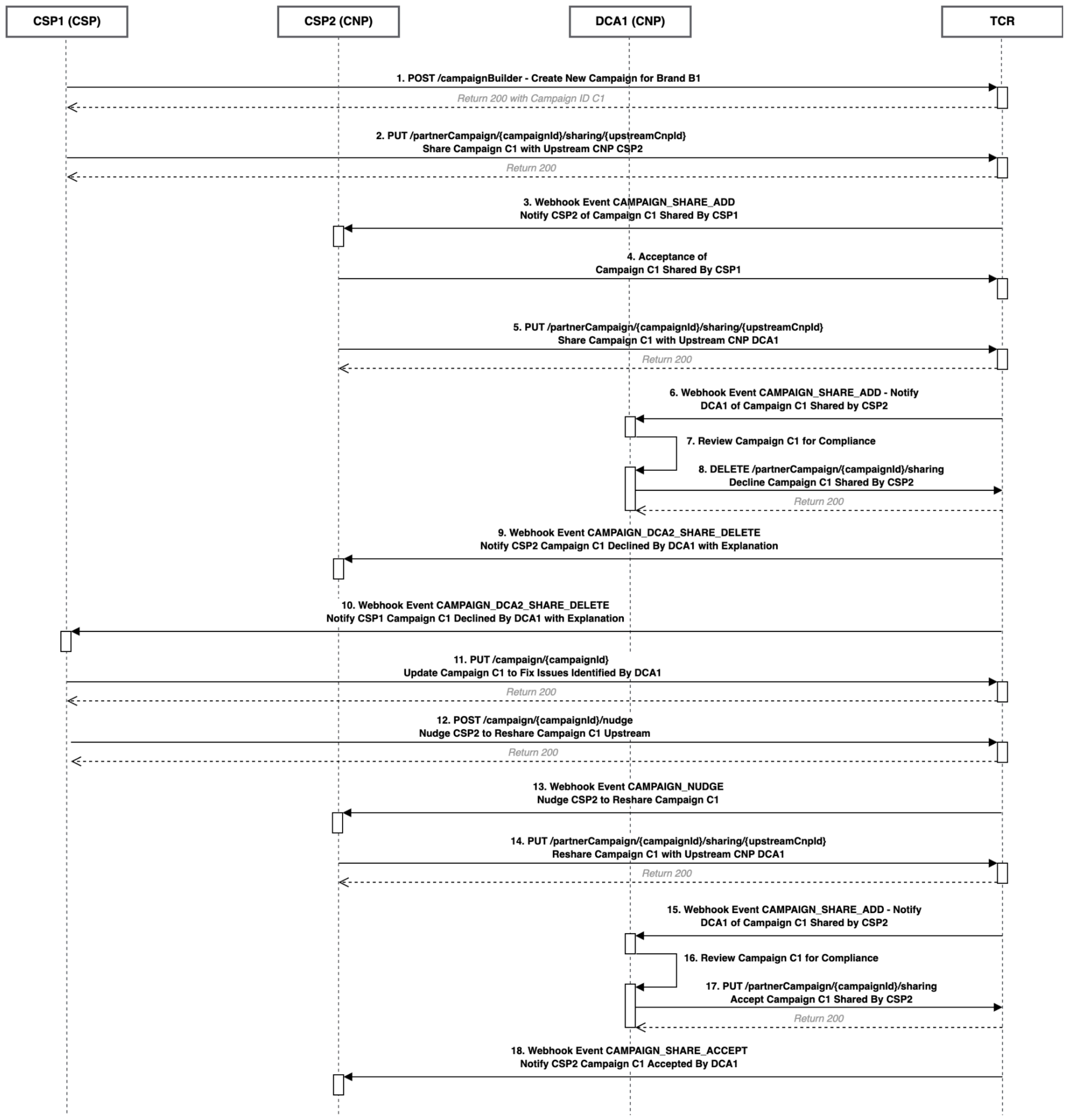
5. When a campaign is shared with an upstream CNP, campaign details are shared based on the role of the CNP. When shared with another CSP, the majority of the campaign details are obfuscated or hidden. However, when a campaign is shared with a DCA, all the campaign details, including the connectivity chain, are shown to the DCA.

Declined Campaigns

It's important to realize that CNPs and DCAs have a responsibility to review every campaign submission for compliance. If a campaign is declined by an upstream CNP, the originating CSP (as well as other downstream parties in the connectivity chain) is notified via a CAMPAIGN_SHARE_DELETE webhook event that identifies the campaign and the reasons for rejection.

If a campaign is declined, the CSP can choose to share the campaign with a different upstream CNP or fix any problems with the campaign and reshare it to the same CNP. When resubmitting a campaign for review, the originating CSP should use the nudge endpoint (see below) to inform the upstream CNP that they have corrected the problem and are appealing the rejection.

Figure 6. *Campaign workflow showing declined sharing events*


CSP1 (CSP)
CSP2 (CNP)
DCA1 (CNP)
TCR
1. POST /campaignBuilder - Create New Campaign for Brand B1
Return 200 with Campaign ID C1
2. PUT /partnerCampaign/{campaignId}/sharing/{upstreamCnpId} Share Campaign C1 with Upstream CNP CSP2
Return 200
3. Webhook Event CAMPAIGN_SHARE_ADD Notify CSP2 of Campaign C1 Shared By CSP1
4. Acceptance of Campaign C1 Shared By CSP1
5. PUT /partnerCampaign/{campaignId}/sharing/{upstreamCnpId} Share Campaign C1 with Upstream CNP DCA1
Return 200
6. Webhook Event CAMPAIGN_SHARE_ADD - Notify DCA1 of Campaign C1 Shared by CSP2
7. Review Campaign C1 for Compliance
8. DELETE /partnerCampaign/{campaignId}/sharing Decline Campaign C1 Shared By CSP2
Return 200
9. Webhook Event CAMPAIGN_DCA2_SHARE_DELETE Notify CSP2 Campaign C1 Declined By DCA1 with Explanation
10. Webhook Event CAMPAIGN_DCA2_SHARE_DELETE Notify CSP1 Campaign C1 Declined By DCA1 with Explanation
11. PUT /campaign/{campaignId} Update Campaign C1 to Fix Issues Identified By DCA1
Return 200
12. POST /campaign/{campaignId}/nudge Nudge CSP2 to Reshare Campaign C1 Upstream
Return 200
13. Webhook Event CAMPAIGN_NUDGE Nudge CSP2 to Reshare Campaign C1
14. PUT /partnerCampaign/{campaignId}/sharing/{upstreamCnpId} Reshare Campaign C1 with Upstream CNP DCA1
Return 200
15. Webhook Event CAMPAIGN_SHARE_ADD - Notify DCA1 of Campaign C1 Shared by CSP2
16. Review Campaign C1 for Compliance
17. PUT /partnerCampaign/{campaignId}/sharing Accept Campaign C1 Shared By CSP2
Return 200
18. Webhook Event CAMPAIGN_SHARE_ACCEPT Notify CSP2 Campaign C1 Accepted By DCA1

Nudging an Upstream CNP

If a campaign is taking an extended period of time to be approved, a CSP can send a notification to the upstream CNP to take action. This notification is referred to as a "nudge" and is invoked through the `POST /campaign/{campaignId}/nudge` endpoint. When using this endpoint, the CSP provides a `nudgeIntent` property in the request body. Possible values include:

- **APPEAL_REJECTION**: This informs the upstream CNP that a campaign has been resubmitted to them for review.
- **REVIEW**: This tells the upstream CNP that the submitted campaign has not yet been reviewed, and is a request for them to take action.

Assigning Phone Numbers

The netnumber Services Registry (nnSR) is a third-party platform that provides a registry of phone numbers associated with messaging campaigns. TCR integrates with the nnSR platform to provide information about a campaign and its status.

> ✏️ **Note:** *TCR is not responsible for phone number provisioning. nnSR authorizes third parties (such as CNPs) to associate telephone numbers with messaging campaigns and provides the tools to do so.*

When a campaign is registered, TCR generates a Campaign ID and sends a message to nnSR with the following information:

- **Campaign ID**: The unique campaign identifier.
- **Status**: A value that indicates the overall status of a campaign (active or inactive in TCR) and which MNOs are currently serving the campaign.
- **Message Class (optional)**: This is only sent if the campaign chooses to route messages to AT&T.
- **Brand ID**: The unique brand identifier.

Registering campaigns in nnSR is independent of a campaign's review and approval. CNPs can start assigning 10DLC telephone numbers in the nnSR platform immediately upon receiving the campaign_id property from TCR. As a campaign runs, any changes to its status with MNOs (e.g., suspensions) will trigger TCR to change the status value in nnSR's registry.

CSPs can use the following endpoints to get more details on the information shared with nnSR:

- `GET /campaign/{campaignId}/osr/attributes`: Gets the campaign attributes shared with nnSR.
- `GET /campaign/{campaignId}/osr/phones`: Returns a list of provisioned phone numbers assigned to the campaign.

CNP Migration

CNP Migration is a feature that lets CSPs or CNPs change upstream connectivity partners for their campaigns. It covers scenarios where, due to changing business relationships, a CSP may want to migrate a campaign from one upstream CNP to another, choose a different DCA entirely, or go directly to the DCA.

In order to use CNP Migration, both the initiating CSP/CNP and the receiving CNP must have the CNP Migration feature enabled on their accounts. The `GET /settings/cnpMigration` endpoint will let these users determine if the feature is enabled. The `PUT /settings/cnpMigration` endpoint allows CSPs or CNPs to change these settings.

> ✏️ **Note:** *If a DCA is involved in the migration, they will also need to have the CNP Migration feature enabled on their account.*

The full details on how to perform a migration, supported scenarios, and the associated endpoints are covered in the Campaign CNP Migration Tool documentation.

Platform Free Trials

If a CSP wants to offer trial accounts to their messaging customers, they can make use of TCR's Platform Free Trial (PFT) campaigns. PFTs allow a brand to send a limited number of messages over several days in order to try the CSP's service. CSPs are required to file monthly reports with TCR on PFT campaign activities.

> ✏️ **Note:** *The PFT feature requires a CSP to submit an application to TCR before they can create a PFT campaign. Please contact TCR Support to request an application (support@campaignregistry.com).*

A CSP can check if the PFT feature is active on their account by using the **`GET /csp/profile`** endpoint. A **`platformFreeTrialEnabled`** attribute will return either true or false.

Registering PFT Campaigns

Following approval, a CSP will be assigned a PFT-enabled brand by the TCR team. CSPs can retrieve the Brand ID and other attributes by using the **`GET /pft/brand`** endpoint.

Additional business rules surrounding PFT campaigns include:

- Only a single PFT-enabled brand is assigned to a CSP account.
- All PFT campaigns must be registered with the use case TRIAL.
- A CSP is allowed to register one PFT campaign for its own use plus one campaign per ISV/reseller.
- When registering a PFT campaign for an ISV, the ISV must be identified in the resellerId attribute in the **`POST /campaignBuilder`** API request.

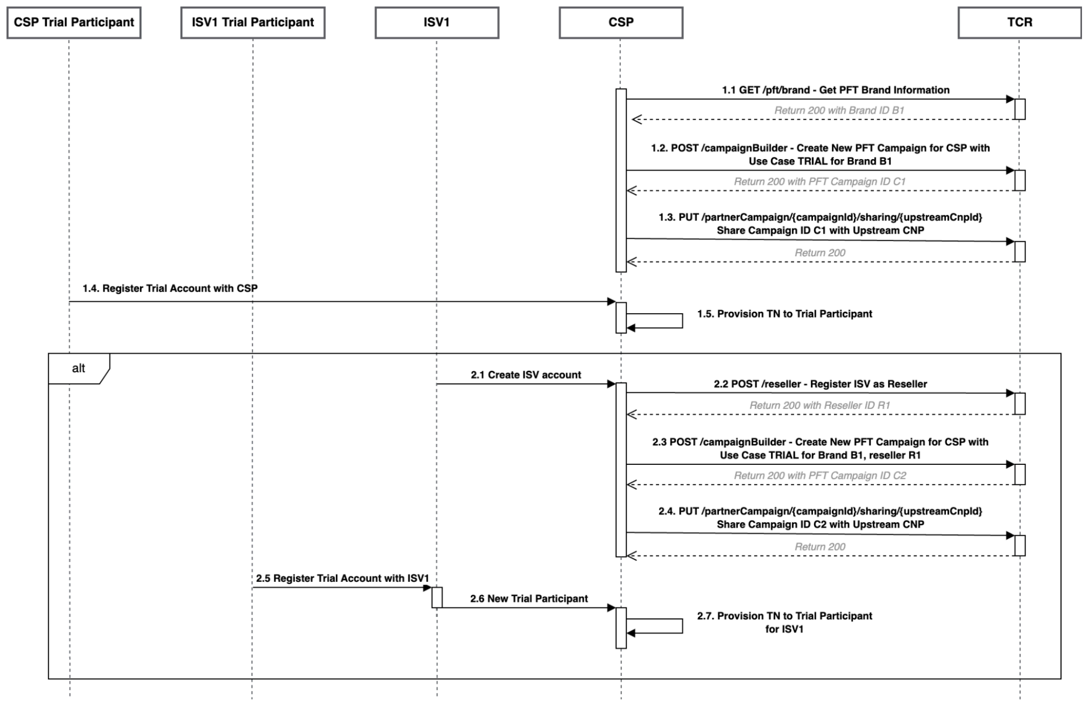
Other than the business rules listed above, the registration and sharing processes for PFT campaigns are identical to standard campaigns.

Reporting PFT User Provisioning

MNOs require CSPs to report PFT user-provisioning events to TCR. A user-provisioning event is recognized when the CSP (or a CSP's ISV) assigns a 10DLC long code to a Platform Free Trial user account. The CSP must report these events on a monthly basis as trial account users are assigned 10DLC long codes. Deactivation or de-provisioning of Platform Free Trial user accounts do not need to be reported. For a sample of the monthly report, please download the Platform Free Trial (PFT) Example Report.

The following diagram shows the process of the PFT lifecycle, from requesting activation of the feature to registering campaigns.

Figure 7. Platform Free Trial workflow.


CSP Trial Participant
ISV1 Trial Participant
ISV1
CSP
TCR
1.1 GET /pft/brand - Get PFT Brand Information
Return 200 with Brand ID B1
1.2. POST /campaignBuilder - Create New PFT Campaign for CSP with Use Case TRIAL for Brand B1
Return 200 with PFT Campaign ID C1
1.3. PUT /partnerCampaign/{campaignId}/sharing/{upstreamCnpId} Share Campaign ID C1 with Upstream CNP
Return 200
1.4. Register Trial Account with CSP
1.5. Provision TN to Trial Participant
alt
2.1 Create ISV account
2.2 POST /reseller - Register ISV as Reseller
Return 200 with Reseller ID R1
2.3 POST /campaignBuilder - Create New PFT Campaign for CSP with Use Case TRIAL for Brand B1, reseller R1
Return 200 with PFT Campaign ID C2
2.4. PUT /partnerCampaign/{campaignId}/sharing/{upstreamCnpId} Share Campaign ID C2 with Upstream CNP
Return 200
2.5 Register Trial Account with ISV1
2.6 New Trial Participant
2.7. Provision TN to Trial Participant for ISV1

Mock Brands and Campaigns

Mock brands and campaigns are available to CSPs in order to test their own applications without interacting with external partners or incurring billable charges. Make note of the following limitations:

- Mock brands and campaigns are only visible to the CSP who created them.
- Mock brands can access any use case type.
- Campaigns created from mock brands are automatically designated as mock campaigns.
- Mock campaigns are not truly functional. A CSP cannot run SMS traffic from mock campaigns.
- Mock campaigns cannot be shared to upstream CNPs and are not registered in the netnumber Services Registry.
- Webhook events associated with mock brands or campaigns are identified with the attribute `mock=true`.

To create a mock brand, use the **`POST /brand/nonBlocking`** endpoint and set the mock attribute to true. For information on EINs available for testing, refer to the CSP Non-Production Integration Testing with Mock EINs document.

Using the CSP API

The CSP API offers both pull and push models of communication between the CSP and TCR. Depending on a CSP's requirements, they can choose to implement one or both. The pull model utilizes API calls for querying or updating TCR entities, while the push model uses webhooks to notify CSPs of events or updates occurring in TCR. For example, a CSP could pull campaign details by calling the **`GET /campaign/{campaignID}`** endpoint and have notifications pushed to them via a webhook when a new campaign or brand is registered in TCR.

API Environments

TCR utilizes several environments for development, testing, and deployment of the code that makes up TCR's products. In addition to a live production environment, TCR maintains additional environments for internal testing and for testing by customers. Customers should utilize the Staging environment to test their own integrations prior to making them live. Staging is the most stable environment of TCR's non-production environments. While access to QA can be requested by customers in order to test new features, it's important to note that it contains code that is in the process of being tested, meaning that some features may not work as expected. Outside of development, the following environments are available to customers:

- **QA**: An internal testing environment where new features are deployed a week prior to release. QA is a dynamic environment that is not as reliable as Staging since it's used for internal testing and addressing any found issues prior to release. Integrations using the QA environment will not impact customer operations.
- **Staging**: An environment where ready-to-release features are deployed a day before product release. This is a stable environment that customers can use to test new features and their own integrations. Integrations using the Staging environment will not impact customer operations.

- **Production**: The live production environment for TCR's products and customers. Integrations using the Production environment will impact live data and customer operations.

API Keys for Environments

API keys are used to authenticate requests and can be acquired by clicking on the Integrations tab in the CSP portal for the environment you are using. In other words, API keys created in the Staging CSP portal will only work in the Staging CSP API environment. Keep this in mind as you develop your integrations.

API Version and Changelogs

The CSP API version number can be retrieved via the **`GET /version`** endpoint. For information on updates and changes, view the changelog. For questions, contact TCR Support.

API Authentication

CSP API REST operations are only accessible via HTTPS to ensure that any data transferred between TCR and the CSP's application is encrypted. To protect against unauthorized access, TCR requires call-level authentication of the CSP's identity using Basic Authentication. Basic Authentication is an industry standard for enforcing access control to HTTP resources.

CSP account managers are responsible for generating APIKEY and SECRET authentication tokens from the Integrations page in the CSP portal. These tokens are passed as User and Password elements in the basic authentication header for the appropriate environment. The following example shows placeholders in a call to retrieve all the available entity types on the TCR platform, where SERVER identifies the environment:

```
$ curl --user [APIKEY]:[SECRET] GET
"http://[SERVER]/v2/enum/entityType" -H "accept: application/json"
```

API Rate Limiting

The CSP API utilizes rate limits to restrict the number of API calls a CSP account can make within a given time period. If this limit is exceeded, CSP API requests will fail with an HTTP status code of 429. Unless specified otherwise, all GET operations are rate-limited at 20 requests per second, while all PUT, POST, and DELETE operations are rate-limited at 20 requests per 10 seconds.

Webhooks and Event Notifications

The CSP API offers webhook events grouped into several categories. CSPs can subscribe to these webhooks to receive a notification of events by category type. A subscription is created by associating a CSP webhook endpoint to a specified event category (see the example below). The webhook event itself is a flat-map JSON data structure that contains an `eventType` property (along with others) that provide important information.

Please be aware that new properties may be added in future releases of the CSP API. To ensure forward compatibility, CSPs should make sure that their webhook endpoints are capable of handling unexpected properties.

Property	Description	Required	Data Type
eventType	Webhook event type	Yes	String
cspId	CSP ID	No	String
cnpId	CNP ID	No	String
cspName	CSP display name	No	String
brandId	Brand ID	No	String
brandName	Brand display name	No	String
brandReferenceId	Brand reference ID (visible to the brand creator)	No	String
brandIdentityStatus	Updated brand identity status. Applicable to BRAND_IDENTITY_ STATUS_UPDATE event	No	String

Property	Description	Required	Data Type
campaignId	Campaign ID	No	String
campaignReferenceId	Campaign reference ID (Visible to campaign creator)	No	String
dcaId	DCA ID	No	String
dcaName	DCA display name	No	String
description	Description	No	String
evpId	External Vetting Provider ID	No	String
evpName	External Vetting Provider display name	No	String
vettingId	External vetting ID	No	String
mnoId	Mobile Network Operator ID. Developers can use the **`GET /enum/mno`** endpoint to retrieve a list of all MNOs.	No	Long
mnoName	Mobile Network Operator display name	No	String
mock	Event associated with a mock brand if the mock attribute exists and is true.	No	Boolean
nudgeIntent	Nudge intent type. Applicable to the CAMPAIGN_NUDGE event.	No	String

Webhook Example - Notification of campaign suspension by AT&T

```
POST / HTTP/1.1
Host: tcr.requestcatcher.com
Accept: *.*
Connection: keep-alive
Content-Length: 226
Content-Type: application/json; utf-8
X-Registry-Signature: SmqzLvP6uQDM6DD2lfdLqwiJJbo=
{
  "cspId" : "S123ABC",
  "brandName" : "Microsoft",
  "brandName" : "Microsoft",
  "mnoName" : "AT&T",
  "campaignId" : "C123ABC",
  "campaignReferenceId" : "C-XXXXX",
  "brandId" : "B123ABC",
  "brandReferenceId" : "Q-XXXXXXXX",
  "mnoId" : 10017,
  "eventType" : "MNO_CAMPAIGN_OPERATION_SUSPENDED",
  "cspName" : "ACME"
}
```

Webhook Delivery

Delivery Order

The CSP API delivers webhook events in chronological order. For example, a campaign registration might generate the following events for a CSP:

- **CAMPAIGN_ADD**: A campaign was registered.
- **CAMPAIGN_SHARE_ACCEPT/CAMPAIGN_SHARE_DELETE**: An upstream CNP accepted a campaign.
- **CAMPAIGN_DCA_COMPLETE**: A campaign was accepted by all DCAs to which it was submitted.

The order of event delivery is exactly the same as the order in which the events were generated. In order to support this First In, First Out (FIFO) requirement, events are funneled through account-specific queues for delivery. There is only one active delivery worker per account/queue. While the delivery order is guaranteed, duplicate events are possible if there are network or system disruptions.

Webhook Retry Policy

The CSP API tries to ensure that each webhook event is accepted by a CSP's registered webhook endpoint. If a webhook endpoint times out or errors out (e.g., an HTTP 400 response), The CSP API will attempt to retry delivery based on a set schedule (as shown below).

> ❗ **IMPORTANT:** *The CSP API will not move to the next event in the queue until the current pending event is accepted by a CSP's endpoint, or if it is discarded following several retry attempts. A CSP's webhook endpoint should not be designed to reject webhook events that*

aren't needed or recognized. Doing so will trigger additional retry attempts on failed events and prevent the timely delivery of other events in the queue.

Webhook Retry Strategy

Webhook Endpoint Response	Retry	Retry Strategy
2xx	No	N/A
3xx	No	N/A
4xx	Yes	Exponential back-off with expiration
5xx	Yes	Exponential back-off with expiration
Network Error/Timeout	Yes	Linear retry with no expiration

Exponential Back-off Retry Schedule

After 13 failed retry attempts, the event is discarded. Currently, there are no methods for recovering these lost events.

Retry Count	Seconds to Next Retry	Total Seconds Elapsed
1	61	61 (1 min)
2	76	137 (2 mins)
3	141	278 (4.6 mins)
4	361	594 (10 mins)
5	685	1279 (21 mins)
6	1356	2635 (44 mins)
7	2461	5096 (1.4 hours)
8	4156	9252 (2.6 hours)
9	6621	15873 (4.4 hours)
10	10060	25933 (7 hours)

Retry Count	Seconds to Next Retry	Total Seconds Elapsed
11	14701	40574 (11 hours)
12	20796	61370 (17 hours)
13	28621	89991 (25 hours)

Webhook Authentication

To ensure secure communication between the CSP API and subscriber applications, all webhook notifications will have an HMAC-SHA1 signed "X-Registry-Signature" authentication header. To verify the signed signature, follow these steps:

1. Canonicalize the JSON raw content (webhook notification body) into a string. If needed, refer to the JCS defined canonicalization scheme.
2. Build the string to be signed as **`stringToSign = {webhook URL + canonicalized string}`**.
3. Set **`AuthKey = {subscriber's SECRET key}`**.
4. Apply the HMAC-SHA1 algorithm on the **`stringToSign`** with **`authKey`** and get the signedString. For more information, refer to the section "How to Validate a Webhook Signature using HMAC" in the Webhook Instructions for CSP API document.
5. Compare the **`X-Registry-Signature`** value with the **`signedString`** to determine the authenticity of the webhook notification.

Webhook IP Addresses

When creating webhooks for CSP applications, the CSP API uses the following IP addresses per development environment:

- **Production**: `44.199.36.118, 52.20.201.41, 52.73.42.221, 3.220.171.38`
- **QA/Sandbox and Staging**: `3.226.115.168, 3.226.75.246, 34.193.103.127`
- **Production (Disaster Recovery)**: `54.68.153.249, 35.161.237.134, 52.42.9.39, 35.160.22.98`
- **QA/Sandbox and Staging (Disaster Recovery)**: `35.80.199.186, 34.212.238.49, 50.112.82.234`

Disaster Recovery IP addresses are servers in a different physical location than the primary data center, and are used in case the primary data center is rendered unusable.

API Endpoints

To view definitions and supported actions for all endpoints in the CSP API, please visit the online CSP API documentation.