Troubleshooting custom analytics integrations

Skip Ahead

- Overwriting experiment and variation name
  - Example
- Considering isHoldback value
- Redirect experiments and referrer value
- Race conditions

RELEVANT PRODUCTS:

- Optimizely Web Experimentation
- Optimizely Performance Edge

THIS ARTICLE WILL HELP YOU:

- Track down and fix issues stemming from custom analytics integrations

Using custom analytics integrations adds additional code execution for an experiment. When building your own custom analytics integration, you should consider how the product you’re integrating with wants to receive the data and how Optimizely sends it.

In general, when working with custom analytics, there are four factors you should be sure to consider:

1. Overwriting experiment and variation data
2. Not taking `isHoldback` into consideration for experiments running for less than 100% traffic
3. Redirect experiments and referrer value
4. Race conditions
**Overwriting experiment and variation name**

When custom analytics integration is enabled for all experiments, a custom integration will run for every single experiment a user has been bucketed into that is active on the page. This means you’ll be executing the same code multiple times, each time potentially overwriting previous experiment information that was supposed to be sent to the integrated platform. If you were to build your own Google Analytics integration, you would most likely use a custom dimension and send the experiment name and variation to it. However, if you were to hardcode your custom dimension value, each experiment would set that custom dimension’s value to their experiment and variation value. If you had three experiments running on the same page, each one would execute and overwrite the custom dimension value. Whichever one is set last would be sent to Google Analytics, while the other two would be lost.

**Example**

With custom analytics you would create an integration to append the experiment ID and the variation name of that experiment that the user was in. A custom dimension value should look like this when it sets it and sends it to Google Analytics:

```
Custom Dimension 2 : “Experiment 123 - Variation #1”
```

If there were three experiments on the page and each one activated immediately, you would see three different custom dimension values get set, and each time one is set, it overwrites the previous one. Meaning only one experiment would get associated with the custom dimension despite there being three experiments active. The order in which experiments activate isn’t always the same, so there is no way to guarantee which one comes first or last.

If your analytics tool of choice doesn’t allow you to create custom dimensions the way Google Analytics does, it might be better to either collect them all in one string or do the integration outside of custom analytics (Project Javascript might be a better for this).

**Considering isHoldback value**

If an experiment has the total traffic allocation set below 100%, anytime a user is added to the traffic allocation that shouldn’t see the experiment, Optimizely still makes a bucketing decision and marks the `isHoldback` value to true. If this value is true, the user is technically not in the experiment and sees the original variation. But because a bucketing decision is made, the custom analytics code will still run, saying a user is in the experiment when in fact they aren’t. **It is important to only run this code if the `isHoldback` is set to false.**

**Redirect experiments and referrer value**

The custom analytics integration framework already accounts for redirect experiments. In a redirect variation, the custom code will execute once on the redirected page (not on the original, pre-redirect). You can get the experiment ID and variation ID via our `getRedirectInfo()` function.

The referrer value can be determined using the `getRedirectInfo()` function.

```javascript
var redirectInfo = window.optimizely.get('state').getRedirectInfo();
```
See this JavaScript file in our GitHub repository as an example.

**Race conditions**

Because you are now sending additional data to another system, it is important to ensure the third party script/object has loaded and is available. We often see third party analytics scripts load asynchronously, and by default, Optimizely will not wait for it. Then the custom analytics code will run before that script has initialized, causing the integration to fail. You can write an interval or some other method to wait for the third party script to be present on the page before executing the custom analytics integration code.