Cookies and localStorage in the Optimizely snippet

Skip Ahead

Cookies
- optimizelyEndUserId
- optimizelyRedirectData
- optimizelyDomainTestCookie
- optimizelyOptOut

RUM cookies
- AWSELB
- optimizelyRumLB

localStorage

Keys set by Optimizely
- optimizelyData${visitorId}${projectNamespace}${event_queue
- optimizelyData${visitorId}${projectNamespace}${events
- optimizelyData${visitorId}${projectNamespace}${layer_states
- optimizelyData${visitorId}${projectNamespace}${session_state
- optimizelyData${visitorId}${projectNamespace}${variation_map
- optimizelyData${visitorId}${projectNamespace}${visitor_profile
- optimizelyData${visitorId}${projectNamespace}${layer_map
- optimizelyData${visitorId}${projectNamespace}${tracker_optimizely
- optimizelyData${visitorId}${projectNamespace}${pending_events

Origins

RELEVANT PRODUCTS:

- Optimizely Web Experimentation
Optimizely Performance Edge

THIS ARTICLE WILL HELP YOU:

- Familiarize yourself with the first-party cookies used by Optimizely
- Understand when visitors are cooked

Optimizely enables you to uniquely identify visitors, track their actions, and deliver consistent experiences across page loads via persistent visitor-level cookies and localStorage. Each cookie Optimizely sets stores a different set of data; this article explains the purpose and behavior of each cookie Optimizely uses. For information on how Optimizely works with Intelligent Tracking Prevention (ITP), see Intelligent Tracking Prevention and Optimizely.

Optimizely Performance Edge is a lightweight experimentation product that delivers significantly faster performance than previous versions of Optimizely. It does this by relying on a streamlined "microsnippet" which limits the range of available features.

Optimizely Performance Edge and Optimizely Web use cookies the same way, so all information contained in this article applies to experiments created with either version of Optimizely.

Take care not to directly reference the cookies and localStorage keys in code, as the specific data format and the cookies themselves may change over time and break your experiences. Instead, use the JavaScript API for Optimizely Web.

If your Optimizely account uses custom snippets, you might see more than one snippet listed for your projects.

Beginning with iOS 12.2 and Safari 12.1 on macOS High Sierra and Mojave, Intelligent Tracking Protection 2.1 is the new default behavior. All client-side cookies (in other words, cookies set from JavaScript via document.cookie) will have a maximum 7-day expiry period.

Please read Intelligent Tracking Protection and Optimizely for more information.

Cookies

Optimizely's JavaScript snippet uses cookies to store some data in visitors' browsers. In most contexts these are first-party cookies.

Important:

When developing experiences on your site, it's important to use the JavaScript API.

Take care not to reference the cookies and localStorage keys directly, as they may change at any time.

If Optimizely is implemented within a third-party frame (a frame or iframe whose domain differs from the domain of the page), the cookies that it needs to read and write are considered third-party cookies.

These cookies are likely to be rejected in browsers like Safari and Firefox. In Chrome, starting with Chrome 80, such cookies are still supported but only if they explicitly include the Secure; SameSite=None; directives. You can read more about this here.
You can opt into a setting that configures the Optimizely snippet to set its cookies with these directives. Please file a support ticket to enable this setting if you are currently running experiments within a third-party frame.

The following are descriptions of all cookies that Optimizely sets:

**optimizelyEndUserId**

- **What:** Stores a visitor's unique Optimizely identifier. It's a combination of a timestamp and random number. No other information about you or your visitors is stored inside.

- **Example value:** "oeu1383080393924r0.5047421827912331"

- **Expiration:** 6 months from last page visit, or a period of your choice (per the setCookieExpiration and extendCookieLifetime APIs)

**optimizelyRedirectData**

- **What:** After Optimizely has executed a redirect experiment, stores various data from the original page so that Optimizely still has access to it on the new page.

- **Expiration:** 5 seconds

**optimizelyDomainTestCookie**

- **What:** When Optimizely loads a URL, the snippet places the cookie to get the current domain, for the purpose of whether cross-domain syncing is possible. If successful, the cookie is immediately removed.

- **Expiration:** 6 months, but the cookie is removed by Optimizely immediately after the set action is successful.

**optimizelyOptOut**

- **What:** Stores a boolean indicating whether the visitor has opted out of participating in Optimizely-powered experimentation. A visitor can opt out by using the opt-out tool [here](#). An Optimizely customer may also set this cookie's value via the snippet's "optOut" API call.

- **Example values:** "true", "false"

- **Expiration:** 10 years from the use of the opt-out tool or snippet's "optOut" API.

In Optimizely, you can use our JavaScript API to set cookie domain. You can also use the API to set cookie expiration and to enable or disable the extend cookie lifetime function.
RUM cookies

For a small portion of your visitors (less than 10%), Optimizely will attempt to collect performance and product usage telemetry; no personal data is collected. We call this process "real user monitoring", or RUM. Visitors who are sampled for this telemetry data will have two cookies on the domain rum.optimizely.com. These are both session cookies; they are not persisted on the visitor's device. Finally, the "domain" attribute of these cookies is always rum.optimizely.com; the aforementioned set cookie domain API has no effect on these.

AWSELB

- **What:** Enables "sticky sessions", which is a requirement of our RUM service. For more information, see [AWS load balancer documentation](#). Its value is random data.
- **Example value:** "799B7FFB1A5DA8D5E09A5D5213B368BD54D987883D4B3B3193567422FA84E59605DE048B6A4B6B488C4DAEFC462F7170F3506CF8FB2F0CBDAA360D722F3F52CFE3A55DF331A5DE690D18747252F1146EF3D3629B00"
- **Expiration:** N/A; it is a session cookie.

optimizelyRumLB

- **What:** Controls the AWSELB cookie's attributes (e.g., SameSite and Secure). For more information, see [AWS documentation on application-controlled session stickiness](#). Its value is always "1".
- **Example value:** "1"
- **Expiration:** N/A; it is a session cookie.

Date created: 6/3/2016

Technical skills needed to implement this solution: Familiarity with browser developer tools

What's the problem?

3rd party cookies are not present within the Resources tab of the browser developer tools.

Solution:

Instead of using the Resource tab within the browser developer tools, you'll have to look into the Network panel for a request sent as part of a goal conversion.

1. Open up the browser developer tools --> Network
2. Fire off Optimizely request
3. Click on the request --> Headers --> Request Headers --> Look for the strings `bucket_map` and `end_user_id`
   1. 
4. Alternatively, you can click on the Request --> Cookies and look for the strings `bucket_map` and `end_user_id`

Why is this information internal-only, and not customer facing?
This is not really information that needs to be digested by a large customer base. 3rd party cookies are more for cross-domain tracking (which works inconsistently), as well as reconciling user actions within our Backend.

---

**localStorage**

Optimizely stores data under the following keys in localStorage.

**Keys set by Optimizely**

The following are keys set by Optimizely Web Experimentation, Personalization, and Optimizely Performance Edge.

- `{visitorId}` is dependent on the visitor ID, generally stored in the optimizelyEndUserId cookie, and `{projectNamespace}` may be a project ID or its account ID depending on whether the project is linked.

**Important:**

When developing experiences on your site, it's important to use the JavaScript API. Here are resources for Optimizely Web.

Take care not to reference the cookies and localStorage keys directly, as they may change at any time.

The 'visitorId' part of the keys below is a combination of a timestamp and random number. No other information about you or your visitors is stored inside.

- `optimizelyData$${visitorId}$${projectNamespace}$event_queue`
  - **What:** Stores event instances that are waiting to be added to the previous $events key.

- `optimizelyData$${visitorId}$${projectNamespace}$events`
  - **What:** Stores event instances that describe actions the visitor has taken on your website.

- `optimizelyData$${visitorId}$${projectNamespace}$layer_states`
  - **What:** Stores bucketing decision (variation and holdback assignment) for each layer in Optimizely Web & Optimizely Personalization.

- `optimizelyData$${visitorId}$${projectNamespace}$session_state`
  - **What:** Tracks the session identifier and timestamp, which helps Optimizely identify sessions for analytics purposes.
optimizelyData$\{visitorId\}$projectNamespace$variation_map

- **What:** Records the variation that the visitor has seen for each experiment. This allows us to deliver a consistent experience on successive page loads.

optimizelyData$\{visitorId\}$projectNamespace$visitor_profile

- **What:** Stores the visitor's values for various audience conditions. This is particularly important for "sticky" conditions like Ad Campaign, Source Type, and Referrer for which we need to rely on the first observed value.

optimizelyData$\{visitorId\}$projectNamespace$layer_map

- **What:** Stores bucketing information within mutual exclusion groups.

optimizelyData$\{visitorId\}$projectNamespace$tracker_optimizely

- **What:** Temporary storage for the batched sending of events to the Optimizely Event API.

optimizelyData$\{visitorId\}$projectNamespace$pending_events

- **What:** Temporary storage of XMLHttpRequest information, enabling the snippet to reliably send requests (e.g., to the Event API) even if the visitor navigates to a new page while a request is in progress.

optimizelyData$\{visitorId\}$events

optimizelyData$\{visitorId\}$event_queue

optimizelyData$\{visitorId\}$layer_states

optimizelyData$\{visitorId\}$session_state

optimizelyData$\{visitorId\}$visitor_profile

optimizelyData$\{visitorId\}$variation_map

These are keys that we used to set, in Classic and/or New Optimizely. The snippet no longer sets data under these keys, for visitors who originally received data for these keys, the snippet is slowly migrating their data over to the new keys.
Origins

Unlike cookie data, localStorage is scoped to a single "origin." An “origin” is a combination of a specific hostname, protocol, and port on your site. Two pages have the same origin if the protocol, port (if one is specified), and host are the same for both pages.

Here’s an example of pages that don’t have the same or different origins as https://shop.example.com:

- http://shop.example.com (different security protocol)
- https://example.com (different subdomain)

You’ll find that the documented localStorage keys are set on every domain that uses Optimizely.

In order for Optimizely to recognize visitors across your pages and deliver consistent experiences, we replicate each visitor’s data on all of your subdomains by routing it through another origin: https://a{yourAccountId}.cdn.optimizely.com/. Your visitors’ data will be set on that origin as well, and on every page load with an Optimizely snippet on it, Optimizely makes an async call to this origin to sync this data. This will sync information across origin as well as across subdomains. Read more about tracking events across domains in Optimizely.

Optimizely does not combine visitor data from multiple accounts.