This article provides technical options to help you understand and mitigate the impact of ITP. You should review these options with your privacy counsel to ensure that you are transparent with your visitors and are complying with requirements in the regions you operate. You should also ensure that your implementation does not circumvent your visitor’s privacy settings, such as cookie settings they may select through your cookie banner. Please refer to this page for an explanation of how to manage Optimizely cookies using our APIs.

Optimizely uses persistent visitor-level cookies and localStorage to uniquely identify visitors, track and attribute their actions to experiments and personalization campaigns, and deliver consistent experiences across page loads and visitor sessions. However, in browsers that use Intelligent Tracking Prevention 2.1 as their default behavior, all client-side cookies—including those used by Optimizely—have a maximum expiration period of seven days.

This means that for visitors who have a gap of seven days or more between site visits, their previous optimizelyEndUserId cookie will not be present when they return, and their existing user identity and state will no longer be available.

If a visitor’s Optimizely cookie is deleted during the course of an experiment or campaign, they will be subject to a new random variation allocation in any experiment or personalization experience for which they meet URL and audience targeting conditions, as shown in the image below and explained in How the Optimizely snippet works: Order of activation.
These visitors’ on-site actions will no longer be counted toward metrics based on initial variation assignment. Instead, their actions will be counted as part of their variation assignment as a net-new visitor or not captured if they do not meet targeting conditions.

**ITP 2.3 - localStorage Expiration**

With the release of ITP 2.3, Safari has begun applying the same expiration periods to localStorage as it does to cookies set client-side. Optimizely depends on localStorage to persist event data for behavioral targeting, some visitor-level attributes and to maintain visitor bucketing for experiments through mid-experiment traffic allocation changes. These functionalities will be constrained to Browsers' persistence capabilities.

ITP 2.3 is currently the default behavior for iOS 13.1 and Safari 13.1 on macOS High Sierra and Mojave, and other
browsers are expected to adopt it in the near future as well.

---

**Optimizely Web Experimentation and Personalization mitigations**

Re-bucketing instances are likely to be limited and will not necessarily result in bias or invalidation of experiment results. However, you can take steps within Optimizely and your implementation to mitigate the impact of ITP on your visitor experience and experiment results.

---

**Mitigations**

The changes introduced as part of ITP 2.1 and 2.2 specifically affect cookies created client-side in JavaScript via `document.cookie`, which is currently the only means for Optimizely Web to manage the cookie-setting process.

One way to ensure the persistence of your cookies, including the `optimizelyEndUserId` cookie, is to manage the cookie creation process server-side at another point in your stack.

**Option 1: Enable and use BYOID**

This can be done by enabling Bring Your Own Visitor ID feature on Optimizely. This feature allows you to define your own visitor ID, either as a cookie, localstorage key, query parameter, or javascript variable. This has several advantages beyond ITP 2.x mitigation, including giving you control over your ID persistence strategy, allowing for a uniform visitor ID across multiple platforms, and reducing cookie bloat. Please see the documentation for further details.

**Option 2: Set optimizelyEndUserId on CDN**

This is not recommended because BYOID is a more complete approach, but another way is to configure cookie creation is through a CDN. This is an option for the UI-based and UI-managed implementation of server-side cookie creation in many cases. Optimizely currently provides documentation for server-side cookie creation through Akamai’s configuration; if you are using another CDN or need guidance on implementing this process in your own stack, contact Optimizely support.

If you are following this process, in addition to the above CDN settings changes, you should also disable the automatic lifetime extension of the visitor ID cookie by running this in the project JS.

```javascript
window["optimizely"].push({
  "type": "extendCookieLifetime",
  "isEnabled": false
});
```

This strategy also has limited functionality when cross-domain tracking is enabled, especially when the different domains follow different strategies for visitor ID persistence.
Results based mitigation

Browser segmentation

Optimizely Web provides out-of-the-box support for filtering your experiment and campaign results by a default set of user segments, including Browser. To validate cross-browser consistency of the results for an experiment, use the Browser segment option to view results that both include and exclude Safari.

Here’s how:

1. At the top of the Results page, click the Segment menu.

2. Select the Match Any Value toggle. This allows an aggregate view of multiple values for a given segment.

3. Select all of the browsers listed in the menu, excluding Safari.

4. Click Apply.

This allows you to view results with all typical reporting figures and statistical significance calculations, with and without the population of visitors who may be affected by ITP 2.1.
localStorage ID persistence

Browser limitations on the lifetime of client-side cookies currently work by overriding the expiration date property of cookies within the browser. localStorage is another way to store information, including Optimizely data, on the client side that is not subject to the current ITP 2.1 expiry limitation. Customers can take advantage of this browser feature to persist client-side cookies and other data in accordance with their general cookie and data policies.

localStorage has a more limited access scope than cookies. In some cases, like when an experiment or campaign spans multiple subdomains or origins, a localStorage-based solution may not prevent a visitor’s user ID from being lost. This setLocalStorage.js code provides an example of code that could be placed in project JavaScript to persist user IDs.