CHROMIUM

BFCache & Cache-control:no-store

Proposed approach

Status	Call for feedback
Last-updated	March 24th 2022

Latest Proposal

Please refer to the public explainer for the latest thinking on BFCache x Cache-Control:No-Store.

Purpose

We would like to share our current plan to make Back/Forward Cache (BFCache) work despite the presence of Cache-control: no-store (CCNS) HTTP headers. Please, have a look, share around, and let us know if there is room for improvements or big concerns.

Chromestatus entry

Context

The CCNS HTTP header is designed to prevent HTTP caches from storing a response.

While this wasn't required by any specification[1], all implementations of BFCache are currently ignoring HTTPS pages with CCNS. This means that many back/forward navigations are much slower than necessary.

Since BFCache has long been a thing, many websites have built a dependency on the unintended interaction between CCNS and BFCaches. In particular, CCNS is used on pages potentially containing personalized details (i.e. logged in state). The expectation vis-a-vis BFCache & CCNS is that if a user logs out from the website, then the previously visited pages can't be restored when another user hits the back button.

Ignoring the BFCache / CCNS context, logout-related inconsistencies can also exist in scenarios involving multiple tabs/windows opened on the same site. Actively checking for login state changes and updating any opened pages accordingly would take care of the problem across the board.

Goals

We would like to:

[1] *Strictly speaking, the specification for the Cache-Control header doesn't apply to BFCaches, because those are not HTTP caches.*

1. Improve back/forward navigations for websites using CCNS...
2. ... while preventing any significant surprises, and...
3. offer better guidance for the future, with the benefit of hindsight.

Timeline (tentative)

1. Heads-up + call for feedback + adjustments if needed.
2. **M100+ (Chrome Canary):** work-in-progress implementation available behind a command line; hands-on + partners checks.
3. **TBD:** Trial run (monitor metrics & feedback from community).
4. **TBD:** Launch or adjust + try again.

Feedback

We welcome feedback on this plan:

- Either as a reply to the PSA.
- Or in a new thread posted to bfcache-dev@chromium.org with **[CCNS]** in the subject line.

Plan

In a nutshell

Our tentative proposal consists of storing CCNS pages in BFCache with the behaviors and carve-outs detailed below.

Note that this is subject to changes, based on feedback from the community, and learnings from experimentation. We also intend to work with other browser vendors and standard bodies on refining & specifying these behaviors, and carve-outs.

Behaviors:

- Evict same-site pages when any HTTP-only cookie changes, including when these cookies expire.
- Evict pages if their HTTP-Authentication state changes.
- Evict same-site pages when the HTTP header Clear-Site-Data includes any of the following values: "cache", “executionContexts”, “*” (to be confirmed with the editors of Clear-Site-Data).

Carve-outs:

In any of the following conditions, CCNS pages will NOT be stored in BFCache:

- When the user has disabled JavaScript.
- When the CCNS page has no HTTP-only cookies. Note: this likely hints at non-standard / non-best practice login schemes, and also covers the cases in which the user has disabled cookies.
- When the behavior is disabled by the administrator of an enterprise/education deployment, via a dedicated group policy. Note: to minimize surprises, the group policy will default to

disabling this new behavior, if any other group policy is set (i.e. requiring no action to preserve the status quo). This defaulting behavior may change in the future depending on what we will learn about these managed environments.

Let us know if you are aware of any significant scenarios that we missed, or for which the proposed plan would create new problems.

Hands-on

An experimental version can be enabled via a command line option:

- --enable-features=CacheControlNoStoreEnterBackForwardCache:level/restore-unless-http-only-cookie-change

References:

- How to launch Chrome with command line options.
- Information about DevTools support for Back/Forward Cache.

Note: this is a work-in-progress implementation, and some aspects are currently missing. Notably, the Clear-Site-Data behavior is not implemented yet.

High level concept

We focused on the idea of using signals indicating that the login state may have changed. The approach is conservative in nature, but based on an experiment we ran, we know that the impact would still be significant.

Let's look at the key authentication means, one by one.

Cookies

The best practice for web authentication is to store a session token in a http-only cookie. This avoids any possibility of extracting the session token via JS and having it leak to an attacker. When a user logs out, the cookie is cleared via an HTTP response.

BFCache will monitor changes to http-only cookies, and evict pages accordingly.

HTTP Basic Authentication

BFCache will drop all the relevant pages, if the HTTP Basic Authentication credentials stored in the browser for those pages are changed while the pages are in BFCache. For example, if the user cleared their stored credentials.

Other schemes

It is possible, but generally not recommended, to implement other authentication schemes. Here are some examples that we have thought about:

- Putting a token in the URL of every page. There is broad agreement that this is very bad practice. As far as we know, the web ecosystem has been moving away from it (e.g. PHP switched to cookies by default in 2009 (PHP 5.3.0)). That said, if cookies are disabled by a user, then this may still be used as a fallback scheme.
- Storing an authentication token in local storage, and manually attaching it to RPCs (e.g. in a single-page app). The presence of an Authorization header could be a signal that we use to retain the current behavior. Feedback on this scenario would be most welcomed.

These schemes could fail to trigger our eviction behaviors or carve-outs, but we believe that this is acceptable since these are rare and discouraged.

Edge cases

JS disabled

Some users may choose to disable JS. This means that pageshow handlers would not fire, and pages unable to do the right thing (see Guidance section). Therefore, for users who have disabled JS, it seems prudent to avoid caching pages with CCNS.

Cookies disabled

Some users may disable cookies. This will likely trigger fallback authentication schemes, making monitoring cookie changes meaningless. In particular, pages with CCNS but no (HTTP-only) cookies, might indicate the use of an alternative authentication scheme in light of cookies being disabled.

Education/Enterprise environments

We know that deployments in Education/Enterprise environments can be tricky to update, and that devices are more likely to be shared (e.g. kiosks, laptops shared between students, etc.).

So, to minimize any risks for these environments, we'll have a group policy to control the behavior of BFCache vis-a-vis CCNS. By default (i.e. when Chrome is running in a managed environment with one or more policies), the behavior will remain unchanged, without the need to set a new policy.

Guidance

We recommend that sites properly handle BFCache by writing pageshow handlers that update the page when it is restored. For instance, the pageshow handler would verify if the user is still logged in before showing the page, and would redirect to a logged out page otherwise.

Alternatively or in addition, we recommend using the HTTP header Clear-Site-Data to clear all caches and other associated state, in response to a user logging out.