Payment Handler Proposals and Discussion Topics

30 March 2020
Web Payments Working Group Virtual Meeting

Overview

- Read all the Proposals

1.1 Explicit Consent for Payment Handler proposal

Proposed: Payment Handlers (both native and web-based) need additional user consent prior to being able to:

- Receive Payment Request events; and
- Access local storage.

Discussion:

- What happens if consent is not given? For example, can the PH ask again in the future?
- Should consent be waived for wellvetted payment apps (browser white list)? In this case, would notification to the user still be useful that a payment app has been installed?

Read the proposal

1.2 Mandatory user interaction with payment handler window

Proposed: Require user consent before any of the following (especially when used together)

- Skip-the-sheet
- Preferred payment handler (user configuration)
- No user interaction (e.g., PH does not open window)

Discussion:

- Opportunities to seek consent (e.g., installation, first usage)

1.3 UX indication for explicit cross-origin context switch

Proposed: Add additional non-blocking signals to the user flow that inform users about the cross origin nature of PHs. Specifically:

- Display a non-blocking alter prior to the PH load.
- Put stronger emphasis on the URL in the PH window.

Not part of proposal:

- Explicit prompt that the user has to confirm before opening the PH window under the hypothesis that this would add too much friction to the flow.
- An onboarding flow under the hypothesis that it would be duplicative of the user consent proposal

Read the proposal

1.4 canMakePayment and hasEnrolledInstrument

Proposed: Define separate events for payment handlers to more clearly align with Payment Request's canMakePayment() and hasEnrolledInstrument() methods. Specifically:

- Both return "true" by default unless any payment handler that is installed returns "false".
- In private browsing mode, no event is sent to the PH so browser always returns "true" for both.

Read the proposal

2.1 Payment handler browser context 3P by default

Proposed:

- By default (and without prior consent), payment handlers will only have 3p access to storage.
- Users can consent to granting 1P storage access to a payment handler (identified by its origin).

Read the proposal

2.2 Read-only storage access before show()

Proposed for Web-based payment handlers:

- Browsers will introduce a “read-only” state for service workers, where:
  - Writing to IndexedDB (the only local storage solution available to service workers) will reject.
  - Fetch requests will reject.
  - Read from IndexedDB will still succeed.
- For a given payment request, all payment handlers start in the “read-only” state.
- The payment handler selected for payment will exit the “read-only” state when the payment request enters “interactive” state.

Read the proposal (including additional bits on Android)

2.2 Read-only storage access before show()

Discussion:

- How well would a “read-only” mode jive well with overall service worker architecture?
- How hard would it be to implement “read-only”?
  - Should the “install” and/or “activate” events in the payment handler also be read-only, if they are preceding a payment event?
  - What if a service worker receives a push notification after “canmakepayment” event, but before it opens a window? Is it possible to start a separate instance of the same service worker in read/write state?
- How critical is freshness?

Read the proposal (including additional bits on Android)

3.1 Skip-the-sheet

Proposed:

- Skip the sheet into any payment handler that supports delegation, even if others are available for the transaction but don't support delegation.

Discussion:

- Does it make sense to skip-the-sheet to a payment handler that supports delegation, even if the user could use another one to pay that does not?
- Does it make sense to skip-the-sheet to a payment handler that is already installed, even if the user could use another one to pay that can be just-in-time installed?
- Today only Chrome implements skip-the-sheet. Should we seek to standardize this behavior?
- Do people expect a prolonged period where a diverse set of payment handlers with different delegation support co-exist in the ecosystem?

Read the skip-the-sheet proposal

3.1 Just-in-Time Installation

Proposed: For a payment method A, crawl for Web-based payment handlers:

- even if other payment handlers for A have been installed, and
- even if the basic card method is requested.

- Discussion:
  - Should we disallow skip the sheet for JIT installable PHs, and always wait for user consent inside the payment sheet/app selector before installing payment handler service workers?
  - For a given transaction, should we limit the number of payment methods that the browser should crawl for? What's the right limit?
  - Similarly, should we limit the number of web app manifests that the browser will crawl for, per payment method manifest?

Read the just-in-time proposal

3.2 Combining Web Authentication and Payment Handler Initiation Gestures

Proposed:

- payment handlers registers credential with browser
- browser changes “Pay” button “Authenticate” and uses one gesture for launching both WebAuthn and payment handler.

- Discussion:
  - This proposal is not likely to work for JIT install as WebAuthn requires an explicit credential enrollment step.
  - Challenge is generated by the browser, which saves a round-trip to the server. Will that work?
  - Question: Are scenarios where this flow should not be followed (e.g., the payment handler is not yet set up for this)?
  - Would it be useful to augment this proposal for other forms of authentication through the Credentials API?

Read the just-in-time proposal