Bring your payment method to the web with the Payment Handler API

By Eiji Kitamura
Developer Advocate in Tokyo

What is the Payment Handler API?

The Payment Request API introduced an open, standards-based way to accept payments in a browser. It can collect payment credentials as well as shipping and contact information from the payer through a quick and easy user interface.

The Payment Handler API opens up a whole new ecosystem to payment providers. It allows a web-based payment application (using an installed service worker) to act as a payment method and to be integrated into merchant websites through the standard Payment Request API.

User experience

From a user's point of view, the user experience looks like this:

1. A user decides to purchase an item and presses the "Buy Now" button on the product detail page.
2. The Payment Request sheet opens.
3. The user chooses a payment method (Payment Handlers have a URL listed below the payment method name).
4. The payment app opens in a separate window where the user authenticates and authorizes the payment.
5. Payment app window closes and the payment is processed.
6. Payment is complete and the Payment Request sheet is closed.
7. The website can display an order confirmation at this point.

Try it yourself here using Chrome 68 beta.

Notice there are three parties involved: an end user, a merchant website, and a payment handler provider.

Merchants' developer experience

For a merchant website, integrating an existing payment app is as easy as adding an entry to **`supportedMethods`** (payment method identifier) and optionally an accompanying **`data`** to the first argument of the Payment Request API. For example, to add a payment app called BobPay with the payment method identifier of **`https://bobpay.xyz/pay`**, the code would be:

```
const request = new PaymentRequest([{
  supportedMethods: 'https://bobpay.xyz/pay'
}], {
  total: {
    label: 'total',
    amount: { value: '10', currency: 'USD' }
  }
});
```

If a service worker that can handle the BobPay payment method is installed, the app will show up in the Payment Request UI and the user can pay by selecting it. In some cases, Chrome will skip ahead to the Payment Handler, providing a swift payment experience!

Chrome also supports a non-standard feature we call just-in-time (JIT) installation. In our example, this would allow BobPay's trusted Payment Handler to be installed on the fly

without the user having visited BobPay's website in advance. Note that the installation only happens after the user explicitly selects BobPay as their payment method within the Payment Request UI. Also note that a payment method (BobPay) can only specify a maximum of one trusted Payment Handler that can be installed just-in-time.

How to build a Payment Handler

To build a payment handler, you'll need to do a little more than just implementing the Payment Handler API.

Install a service worker and add payment instruments

The heart of the Payment Handler API is the service worker. On your payment app website, register a service worker and add payment instruments through **`paymentManager`** under a **`ServiceWorkerRegistration`** object.

```
if ('serviceWorker' in navigator) {
  // Register a service worker
  const registratoin = await navigator.serviceWorker.register(
    // A service worker JS file is separate
    'service-worker.js'
  );
  // Check if Payment Handler is available
  if (!registration.paymentManager) return;

  registration.paymentManager.instruments.set(
    // Payment instrument key can be any string.
    "https://bobpay.xyz",
    // Payment instrument detail
    {
      name: 'Payment Handler Example',
      method: 'https://bobpay.xyz/pay'
    }
  )
}
```

Listen to **`paymentrequest`** events and get user consent for payment

To handle actual payment requests, listen to **`paymentrequest`** events in the service worker. When one is received, open a separate window and return a payment credential after getting the user's authorization for a payment.

```
const origin = 'https://bobpay.xyz';
const methodName = `${origin}/pay`;
const checkoutURL = `${origin}/checkout`;
let resolver;
let payment_request_event;

self.addEventListener('paymentrequest', e => {
  // Preserve the event for future use
  payment_request_event = e;
  // You'll need a polyfill for `PromiseResolver`
  // As it's not implemented in Chrome yet.
  resolver = new PromiseResolver();

  e.respondWith(resolver.promise);
  e.openWindow(checkoutURL).then(client => {
    if (client === null) {
      resolver.reject('Failed to open window');
    }
  }).catch(err => {
    resolver.reject(err);
  });
});

self.addEventListener('message', e => {
  console.log('A message received:', e);
  if (e.data === "payment_app_window_ready") {
    sendPaymentRequest();
    return;
  }

  if (e.data.methodName === methodName) {
    resolver.resolve(e.data);
  } else {
    resolver.reject(e.data);
  }
});

// Get the user's authorization

const sendPaymentRequest = () => {
  if (!payment_request_event) return;
  clients.matchAll({
    includeUncontrolled: false,
    type: 'window'
  }).then(clientList => {
    for (let client of clientList) {
      client.postMessage(payment_request_event.total);
    }
```

```
  });
}
```

Identifying a payment app

To identify the payment app from a URL-based payment method identifier (e.g., `https://bobpay.xyz/pay`), you'll need to include the following declarative materials.

1. A payment method identifier: points to a payment method manifest.
2. A payment method manifest: points to a web app manifest and supported origins.
3. A web app manifest: describes a website that hosts a service worker that handles payment requests.

To learn more about how to implement a payment app, see Quick guide to implementing a payment app with the Payment Handler API.

Example payment methods

Because the Payment Handler API is designed to be flexible enough to accept any kind of payment method, supported methods can include:

- Bank transfers
- Cryptocurrencies
- E-money
- Carrier billings
- Merchant's point system
- Cash on delivery (Merchant's self-served)

Resources

- Introducing the Payment Request API
- Deep Dive into the Payment Request API
- Payment Handler API
- Quick guide to implementing a payment app with the Payment Handler API
- Example payment app - BobPay

Last updated July 2, 2018.