# Credential Management API Feature Detection Check-up

**By** Eiji Kitamura
Developer Advocate in Tokyo

## TL;DR

WebAuthn helps increase security by bringing public-key credential based authentication to the Web, and is soon to be supported in Chrome, Firefox and Edge (with the updated spec). It adds a new kind of `Credential` object, which, however, may break websites that use the Credential Management API without feature-detecting the specific credential types they're using.

## If you are currently doing this for feature detection:

```
if (navigator.credentials && navigator.credentials.preventSilentAccess) {
  // use CM API
}
```

## Do these instead:

```
if (window.PasswordCredential || window.FederatedCredential) {
  // Call navigator.credentials.get() to retrieve stored
  // PasswordCredentials or FederatedCredentials.
}

if (window.PasswordCredential) {
  // Get/Store PasswordCredential
}

if (window.FederatedCredential) {
  // Get/Store FederatedCredential
}

if (navigator.credentials && navigator.credentials.preventSilentAccess) {
  // Call navigator.credentials.preventSilentAccess()
}
```

See changes made to the sample code as an example.

Read on to learn more.

**Note:** If you are using Google identity as a primary way for your users to sign-in, consider using the one tap sign-up and automatic sign-in JavaScript library built on the Credential Management API. It combines Google sign-in and password-based sign-in into one API call, and adds support for one-tap account creation.

## What is the Credential Management API

The Credential Management API (CM API) gives websites programmatic access to the user agent's credential store for storing/retrieving user credentials for the calling origin.

Basic APIs are:

- `navigator.credentials.get()`
- `navigator.credentials.store()`
- `navigator.credentials.create()`
- `navigator.credentials.preventSilentAccess()`

The original CM API specification defines 2 credential types:

- `PasswordCredential`
- `FederatedCredential`

The `PasswordCredential` is a credential that contains user's id and password.
The `FederatedCredential` is a credential that contains user's id and a string that represents an identity provider.

With these 2 credentials, websites can:

- Let the user sign-in with a previously saved password-based or federated credential as soon as they land (auto sign-in),
- Store the password-based or federated credential the user has signed in with,
- Keep the user's sign-in credentials up-to-date (e.g. after a password change)

## What is WebAuthn

WebAuthn (Web Authentication) adds public-key credentials to the CM API. For example, it gives websites a standardized way to implement second-factor authentication using FIDO 2.0 compliant authenticator devices.

On a technical level, WebAuthn extends the CM API with the `PublicKeyCredential` interface.

## What is the problem?

Previously we have been guiding developers to feature detect the CM API with following code:

```
if (navigator.credentials && navigator.credentials.preventSilentAccess) {
  // Use CM API
}
```

But as you can see from the descriptions above, the `navigator.credentials` is now expanded to support public-key credentials in addition to password credentials and federated credentials.

The problem is that user agents don't necessarily support all kinds of credentials. If you continue feature detect using `navigator.credentials`, your website may break when you are using a certain credential type not supported by the browser.

**Supported credential types by browsers**

| | PasswordCredential / FederatedCredential | PublicKeyCredential |
|---|---|---|
| **Chrome** | Available | In development |
| **Firefox** | N/A | Aiming to ship on 60 |
| **Edge** | N/A | Implemented with older API. New API (navigator.credentials) coming soon. |

## The solution

You can avoid this by modifying feature detection code as follows to explicitly test for the credential type that you intend to use.

```
if (window.PasswordCredential || window.FederatedCredential) {
  // Call navigator.credentials.get() to retrieve stored
  // PasswordCredentials or FederatedCredentials.
}

if (window.PasswordCredential) {
  // Get/Store PasswordCredential
}

if (window.FederatedCredential) {
  // Get/Store FederatedCredential
}

if (navigator.credentials && navigator.credentials.preventSilentAccess) {
  // Call navigator.credentials.preventSilentAccess()
}
```

See actual changes made to the sample code as an example.

For a reference, here's how to detect **`PublicKeyCredential`** added in WebAuthn:

```
if (window.PublicKeyCredential) {
  // use CM API with PublicKeyCredential added in the WebAuthn spec
}
```

## Timeline

Earliest available implementation of WebAuthn is Firefox and is planned to be stable around early May 2018.

## Finally

If you have any questions, send them over to @agektmr or agektmr@chromium.org.

*Last updated July 2, 2018.*