From Last Time:

Straw-man solution: CrossOriginStorageAccessCredential:

Semantically: Holding this object allows you to use cross-origin cookies for the IDP. Each Credential is bound to an IDP-RP pair.

How it would work: if one is held, it can be Stored, and if it's been Stored it can be Collected. Creation doesn't happen directly on the RP, only through Discovery.

Discovery is complicated because (1) the IDP needs to be able to opt-in per RP, (2) it may require a redirect away and back mid-request, and (3) this is the point at which we need to show UI to break down the privacy barrier.

Controversial part:

Discovery can only be single-IDP. We give UI that looks like "Try to log in with your account at "foo.com". It must redirect the current navigable to the IDP's origin. Once there, the IDP can see what origin wants to create a credential on them, can Create and Store a Credential. Also during this process the RP should allow silent access to the Credential.

First improvement:

This makes creating a credential only happen amid a redirect-flow. We can improve this by allowing an IDP to pre-register what RPs are acceptable (a list of < M origins, or an endpoint that responds to CORS Origin headers with ACAO). This lets a pre-registered IDP skip the redirect flow.

Magic moment:

Stored COSACs, stored passwords, WebAuthN, and try another site for a COSAC can all be presented with each other.

Code examples:

IDP on auth redirect or login:

```
// Check for pending requests, i.e. redirect flows
for (let r in navigator.credentials.crossSiteRequests.getPending()) {
  if (IDP_DEFINED_ALLOW_ORIGIN(r.origin)) {
    navigator.credentials.crossSiteRequests.allow(r);
  }
}

// Optional: preregister to avoid redirect flow
navigator.credentials.crossSiteRequests.allow({
  "future-requests" : {
    allowlist: ["https://rp1.biz", "https://rp2.info"], // optional
    "dynamic-via-cors": "https://api.login.idp.net/v1/foo", // optional
  }
});
```

RP code to get a credential:

```
// Check for already logged in, silently grab any credentials
let credential = navigator.credentials.get({
  mediation: "silent",
  password: {},
  "cross-site": {
    "allow-redirect": false,
    providers: [
      {
        "auth-origin": "https://login.idp.net",
      },
      // ... as many as you want
      {
        "auth-origin": "https://account.identity.ninja",
      },
    ],
  },
});

// On authenticator selection click, force a dialog
let credential = navigator.credentials.get({
  mediation: "conditional",
  password: {},
  "public-key": {},
  "cross-site": {
    "allow-redirect": true,
    providers: [
      {
        "auth-origin": "https://login.idp.net", // can be derived from "auth-link" if not provided.
        "auth-link": "https://bounce.helper.live/help?url=https://login.idp.net/login-page",
      },
    ],
  },
});

// Optional: On login link click before navigation
let credential = navigator.credentials.get({
  mediation: "conditional",
  password: {},
  "public-key": {},
  "cross-site": {
    "allow-redirect": false,
    providers: [
      {
        "auth-origin": "https://login.idp.net",
      },
      // ... as many as you want
      {
        "auth-origin": "https://account.identity.ninja",
      },
    ],
    // or if SSO, "allow-redirect": true, and only one provider
  },
});
```

RP using that credential:

```
let credential = navigator.credentials.get({});
document.requestStorageAccess({"top-level-access": credential});
// alternatively, credential.allow();
// now all credentialed requests to the "auth-origin" origin in this document
// will have unpartitioned cookies for the rest of this document's lifetime.
```

UI/UX

4 states for a given IDP on a given RP:
1. COSACredential in the Store for this pair, allowing silent access: logged in
2. COSACredential in the Store for this pair, requires UI in credman: logged out, but linked
3. IDP pre-allow called with some arguments: Registered and maybe logged in, but not linked
4. Brand new: requires a login flow with a top-level navigable

UI is in the credential chooser UI (https://w3c.github.io/webappsec-credential-management/#user-mediated-selection).
This is the as-of-yet unrealized UI to let users pick between different login methods.

(A) Log back in with your bar.com account — bar.com's favicon: 🎵

Q: Multiple accounts under 1 IDP?

(B) Link to your account at accounts.foo.com

(1) has no UI if done properly with CredMan, falls back to the same as (2)
(2) has UI (A) as N of >= N options
(3) The same as (4), maybe with a favicon.
(4) has UI (B) as 1 of >= 1 options in the credential chooser UI

Clicking (A) returns a Credential immediately to the page.
Clicking (B) will do the same if pre-allow was done and permits this page.
If this domain isn't allowed, an error will be made clear to the page.
If no pre-allow was done, a top-level redirect will happen.

FedCM Relation

This is fundamentally different:
authenticating via enabling cross-site credentialed requests
vs
sharing a user-legible identity with some properties

clearly a need for something to fill the role of **"easy-to-use top-level cookie access for logins API".**

What is the difference between
"new working mode of IdentityCredential" and "alternative credential" in API space?

If we can build a spec that are these semantics, but with optional fields that each add a capability of FedCM, that might be possible to do. However, this is a major shift in the integration with CredMan, so I don't see piecemeal PRs creating the holistic change we need here. That already failed with a prior iteration of this effort in early 2023.

Open questions that jump to mind:

how do we handle things that don't make sense for the simple mode:
`IdentityCredentialRequestOptionsContext context = "signin";`
how do we handle complicated features like Multi-IDP that aren't needed with this?

Alternatively, we can enable smooth upgrades in separate Credentials with either:
1) navigator.credentials.supported()
   > ["cross-site", "federated", "identity", "password", "public-key"]
   which the RP uses to construct the correct argument.
2) Adding a way for the collected credential list to be pruned down in credential management spec.
   particularly interface types. Permit suppression of one if another is present.