



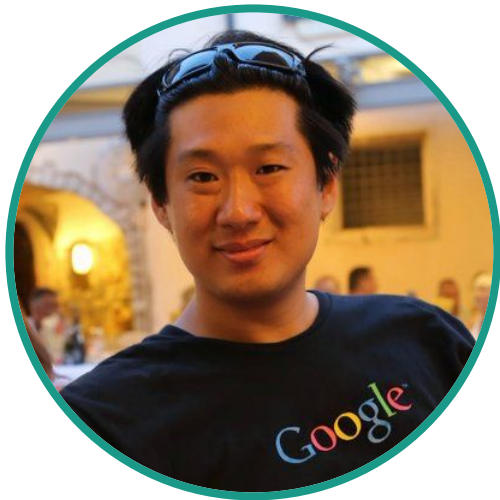
FedCM 101

OAuth Security Workshop 2024

FedCM 101



- State of the ecosystem
- Related solutions
- What is FedCM?
- How does it work?
- Demo
- Status
- What's next?
- Discussion



Sam Goto

Sr. Staff Software Engineer, Chrome
Google

<https://sgo.to>



Tim Cappalli

Sr. Architect, Identity Standards
Okta

<https://timcappalli.me>

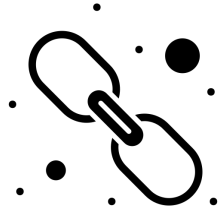
State of the ecosystem

The Problem

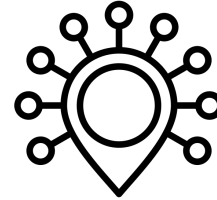


Non-transparent, uncontrollable tracking of users across the web needs to be addressed and prevented, *without breaking it.*

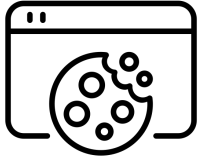
Common Forms of Tracking



Link
Decoration



Network
Addresses



Third-Party
Cookies

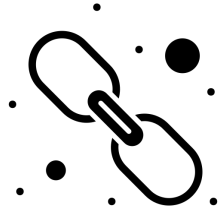


Bounce
Tracking



Fingerprinting

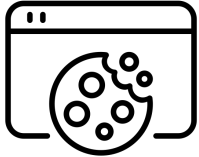
Common Forms of Tracking



Link
Decoration



Network
Addresses



Third-Party
Cookies

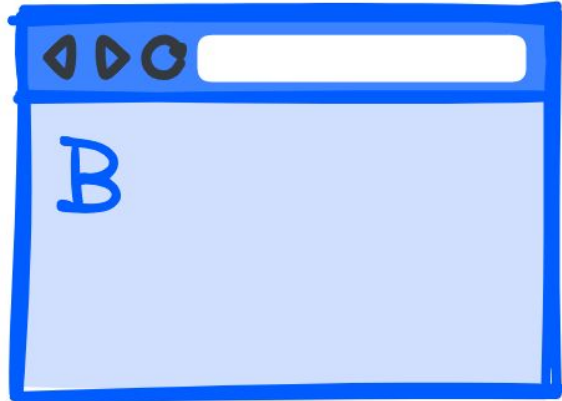


Bounce
Tracking

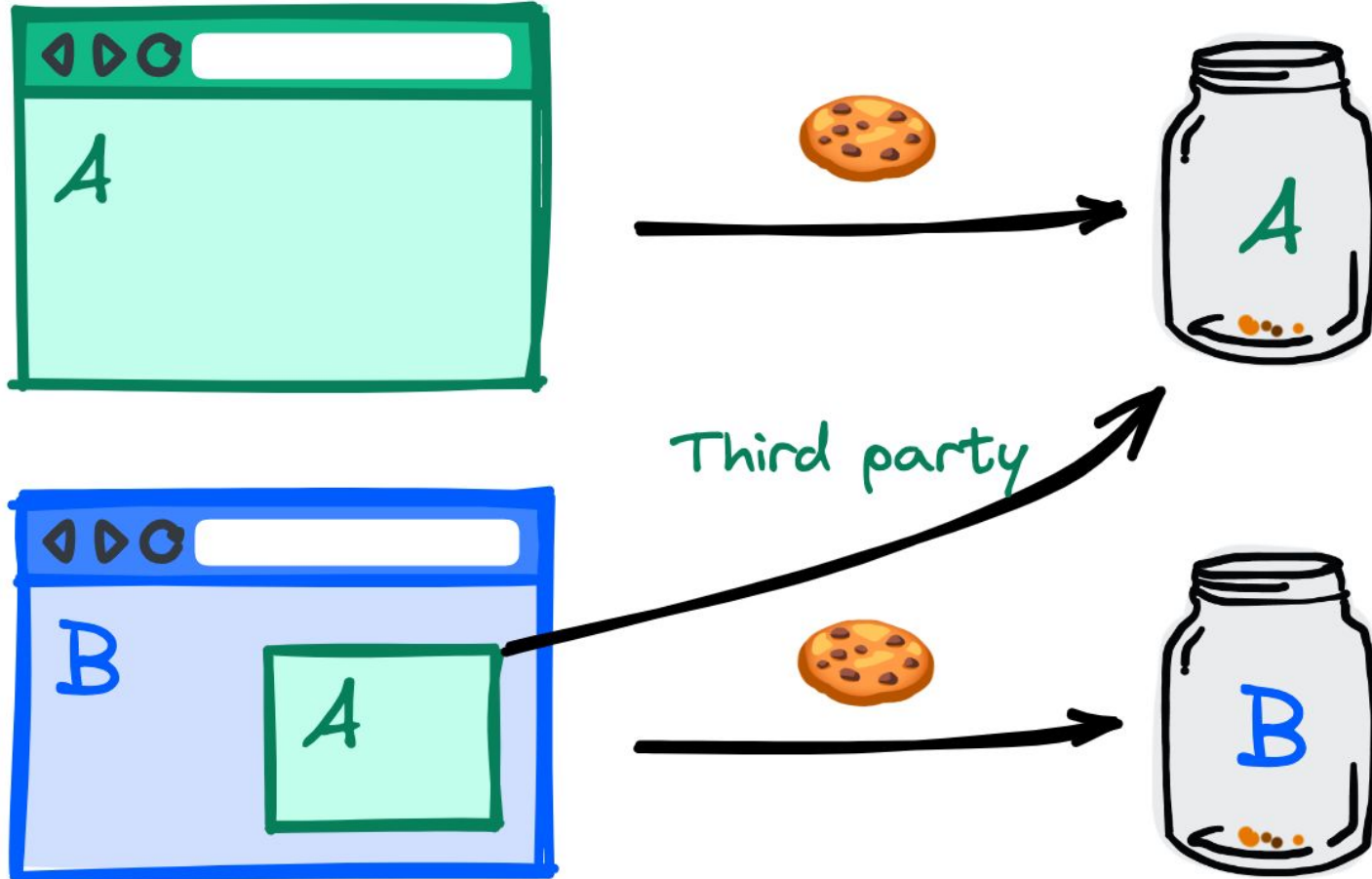


Fingerprinting

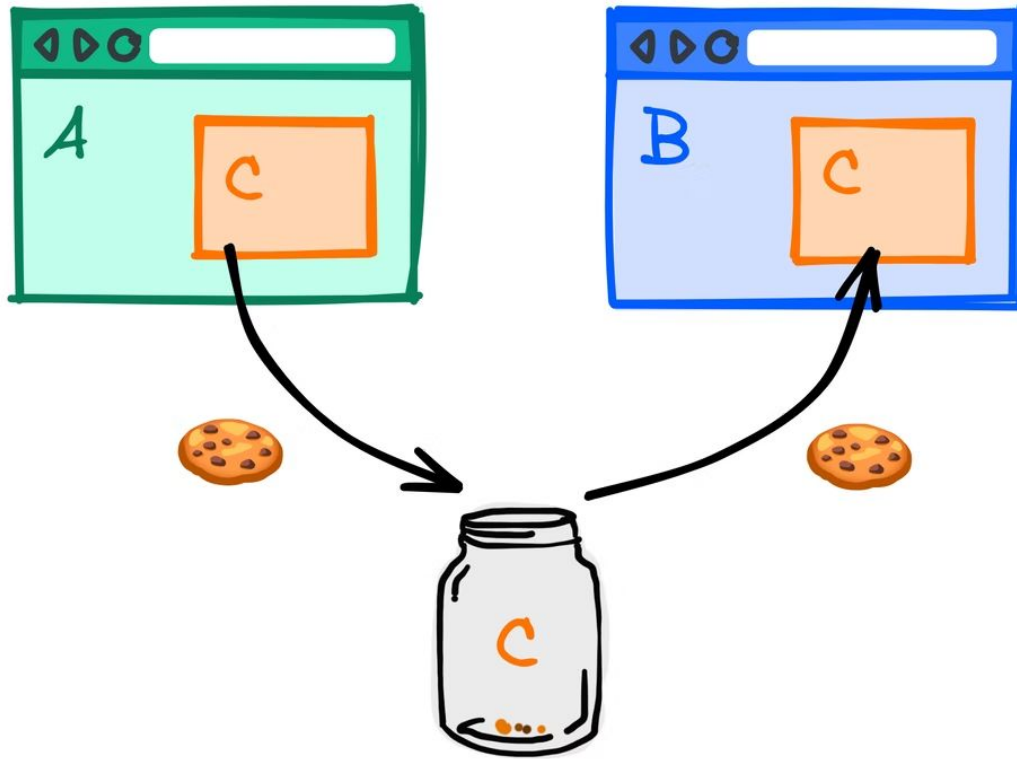
Third Party Cookies



Third Party Cookies



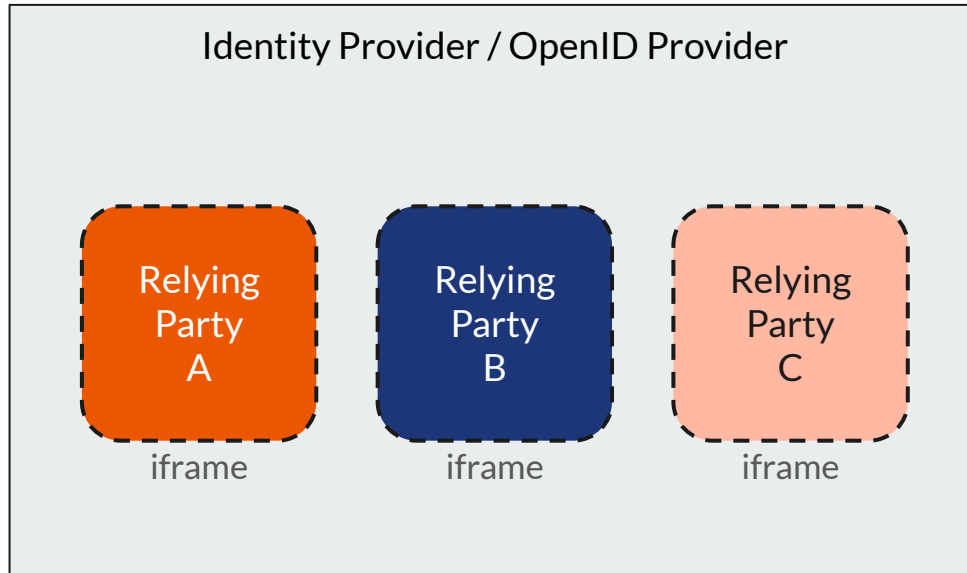
Third Party Cookies



A, B - top-level sites
C - embedded site

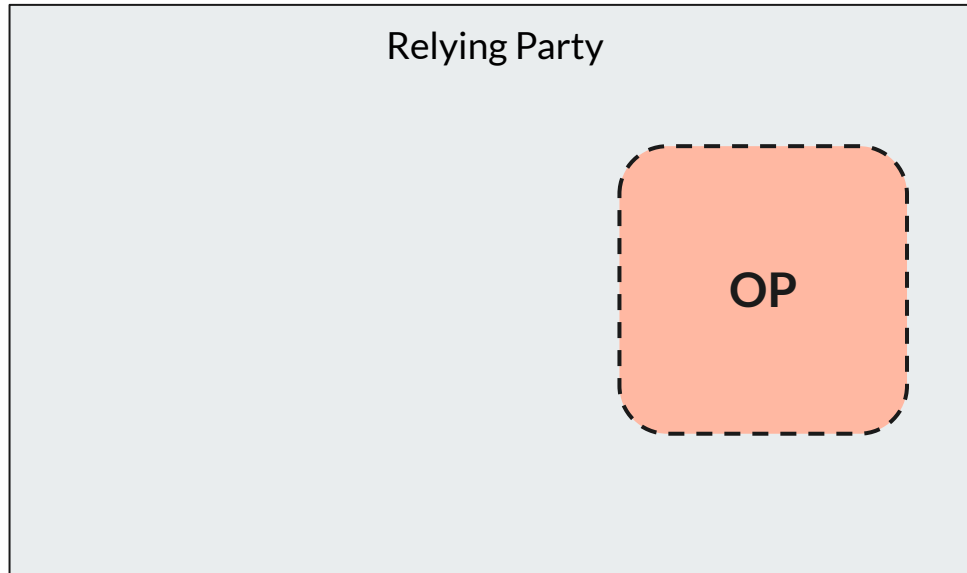
Third Party Cookie Usage for Federation

OIDC Front-Channel Logout



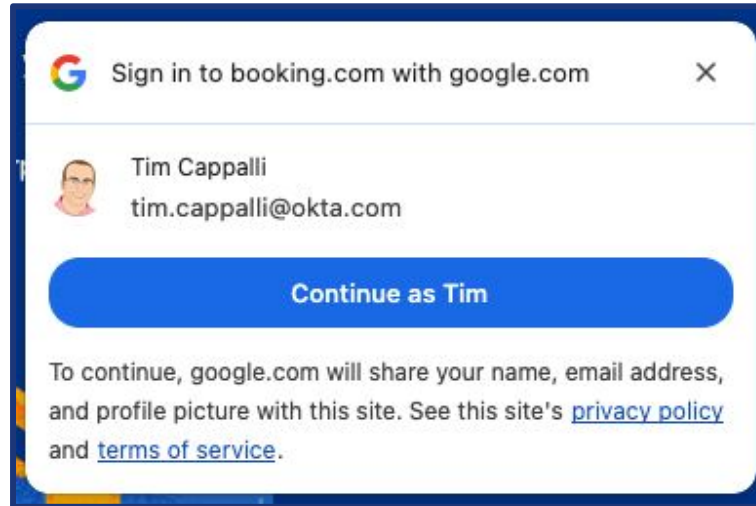
Third Party Cookie Usage for Federation

OIDC Session Management



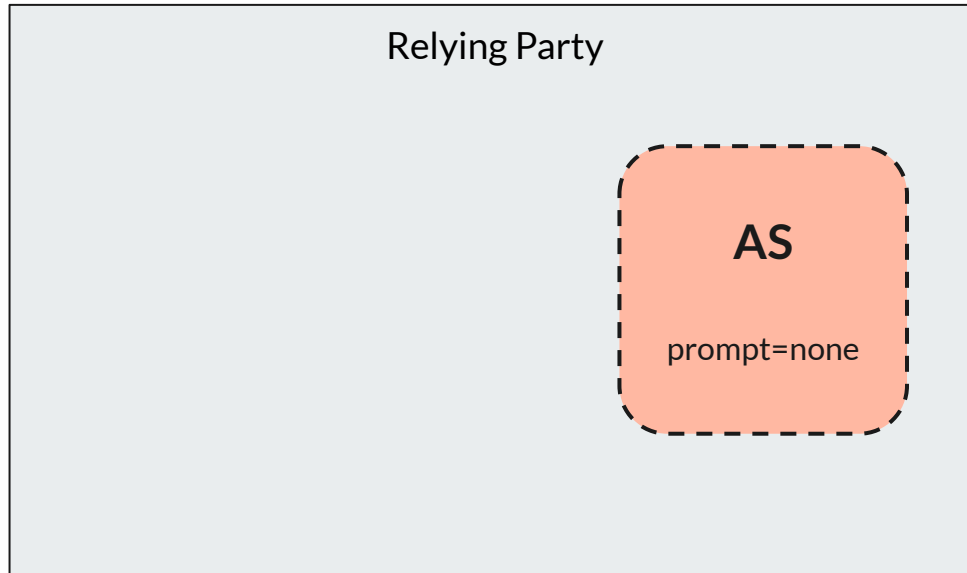
Third Party Cookie Usage for Federation

Iframe-Based Login Widgets

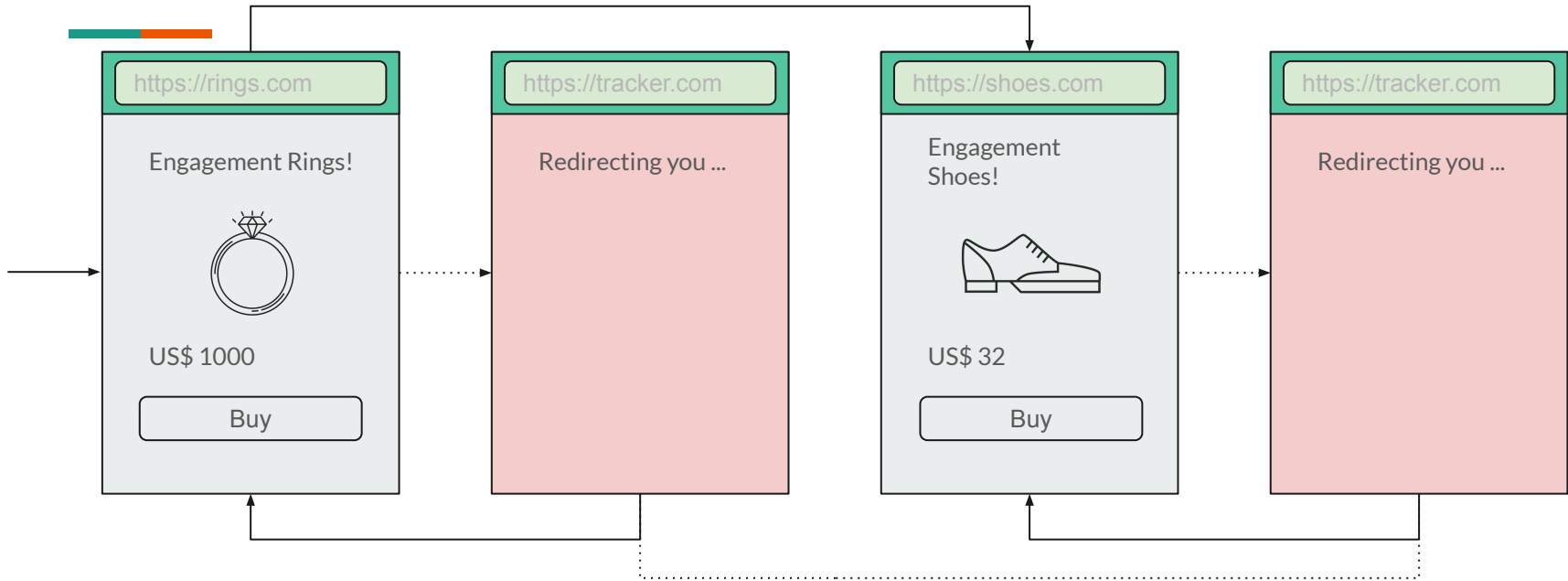





Third Party Cookie Usage for Federation

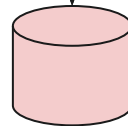
Iframe-Based Token Refresh



Bounce Tracking



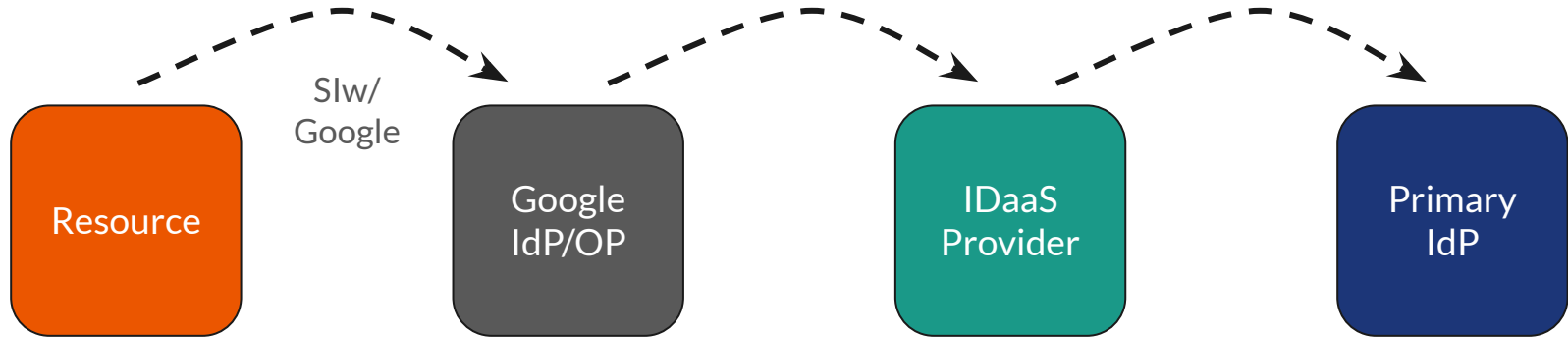
-  Browser
-  RP
-  Tracker



User 123 viewed engagement rings
User 123 viewed engagement shoes
....

OIDC, SAML, and OAuth

Looks like bounce tracking, especially multi-hop federation!



OIDC, OAuth, and SAML

```
https://okta.okta.com/oauth2/v1/authorize
?client_id=okta.16852165-7c64-4cc6-886b-16a497752fcb
&code_challenge=Hxgh7sAwFTrxEgNbHLcLYyGTuo6cYJ
&code_challenge_method=S256
&nonce=J73yyX7Wmnr86BgsCSRejV5hZiMGaKckB7DttbRE
&redirect_uri=https%3A%2F%2Fokta.okta.com%2Fcallback
&response_type=code
&state=WFZDDNYrCSDxPFhJ3RG39e8yH7gMdRUehXxKhGwTtxL
&scope=openid%20profile%20email%20
```

```
https://okta.okta.com/app/google/5aX4dGOo1d8/sso/saml
?SAMLRequest=fVJNT%2BMwEL0j7X%2...
&RelayState=https%3A%2F%2Faccounts...
```

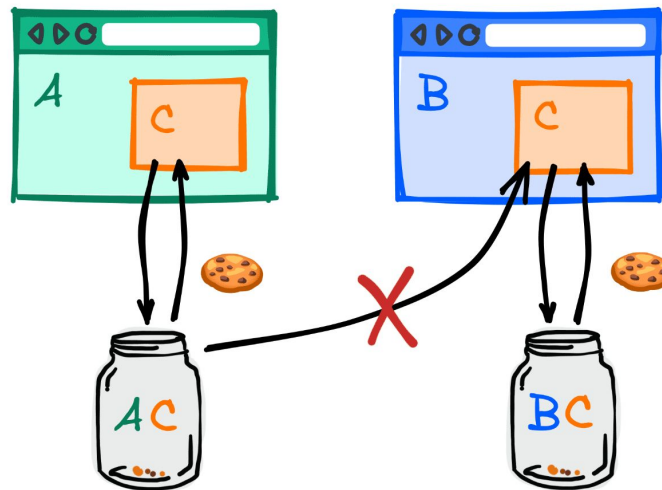
Related solutions

Cookies Having Independent Partitioned State (CHIPS)



Partitioned

cross-site cookies that
are partitioned by
top-level context

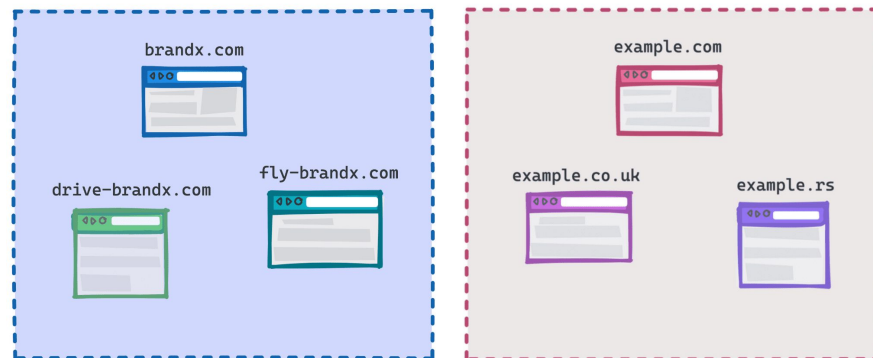


A,B - top-level sites
C - embedded site

Set-Cookie: __Host-name=value; Secure; Path=/; SameSite=None; **Partitioned**;

Related Website Sets (RWS)

- Associate different, but related domains as the same-party
- Relations declared in a public GitHub repo
- Currently only in Chrome
- Previously known as *First Party Sets*



Storage Access API

- allows iframes to request access to unpartitioned cookies
- "escape hatch"
- challenges:
 - prompts are not contextual
 - slight variances in implementations



Do you want to allow "storage-access-api-demo.glitch.me" to use cookies and website data while browsing "storage-access-api-demo-site-b.glitch.me"?

This will allow "storage-access-api-demo.glitch.me" to track your activity.

Don't Allow

Allow

Allow embedded content?



storage-access-api-demo.glitch.me wants to use info they've saved about you

storage-access-api-demo.glitch.me will know that you visited storage-access-api-demo-site-b.glitch.me

[Learn more about embedded content](#)

Block

Allow

What is FedCM and how does it work?

What is FedCM?



The *Federated Credential Management API* (FedCM) provides a **use-case-specific abstraction** for federated identity flows on the web, by exposing a browser mediated experience that allows users to choose accounts from IdPs to login to websites.

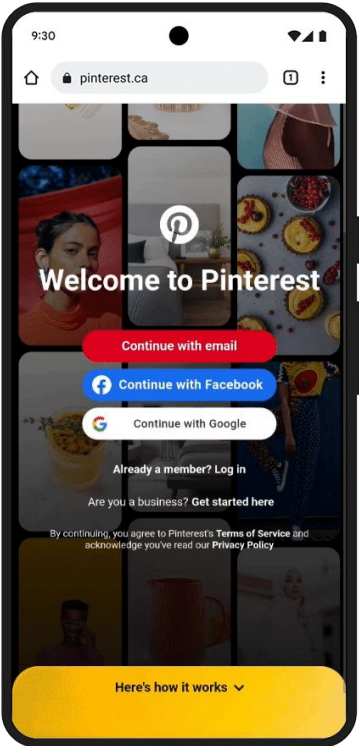
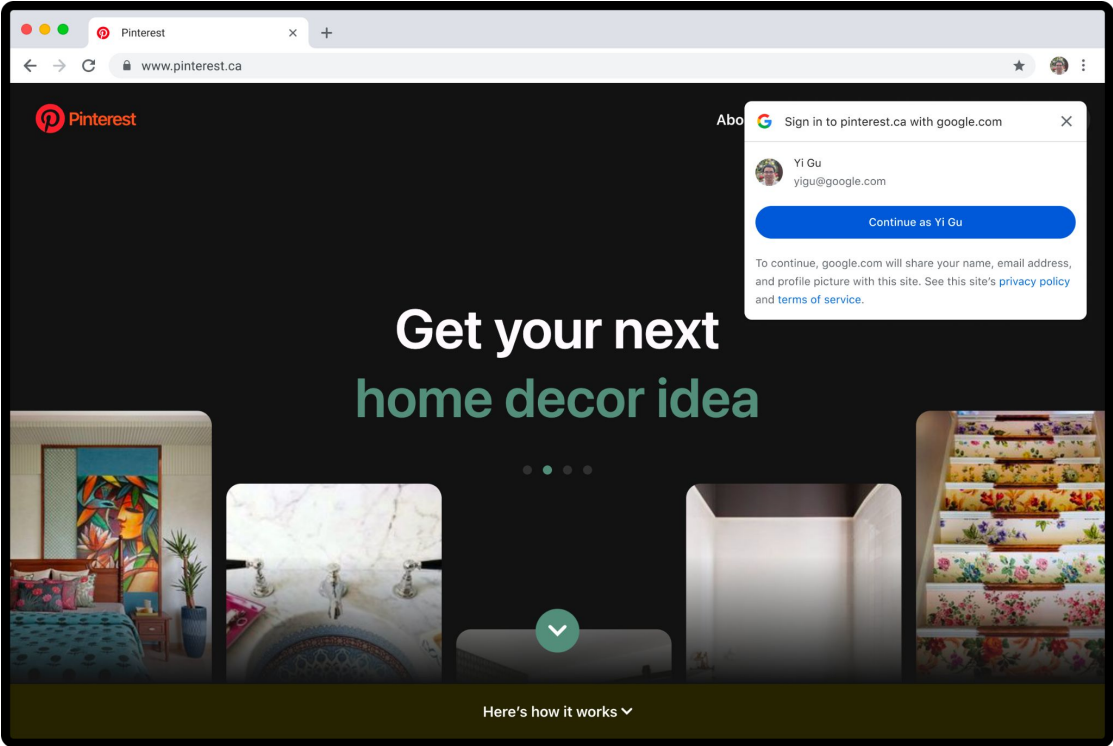
What is FedCM?



1. Identity federation flow starts from the RP
2. The user consents to use the IdP account
3. The browser sends requests to the IdP endpoints directly
4. The browser returns the assertion to the RP

These interactions are done without disclosing who and from where, until the user consents.

What is FedCM?



API Structure



```
const credential = await navigator.credentials.get({
  identity: {
    providers: [{
      configURL: 'https://accounts.idp.example/config.json',
      clientId: 'ec40746e-f61c-48d9-b178-4750d955297c',
      nonce: '4jdz5TmENVTAHmMMDiRHWkj9'
    }]
  }
});

const { token } = credential;
```

Relying Party (RP) Requirements



- Check for FedCM support
- Call `navigator.credentials.get()` instead of initiating a redirect

Identity Provider (IdP) Requirements

- *Well-known* endpoint
- *Config* endpoint
- *Accounts* endpoint
- *Client metadata* endpoint
- *Identity assertion* endpoint
- *Disconnect* endpoint

DESIGN PHILOSOPHY
*put most of the work
on the user agent and IdP,
simplifying deployment for
RPs and simplifying the
experience for users*

Well-Known endpoint

- no cookies, no origin, no referrer
- Well-known used to prevent RP <> IdP correlation prior to consent
- can be challenging to update a well-known at APEX in many organizations

```
GET /.well-known/web-identity HTTP/1.1
Host: idp.example
Accept: application/json
Sec-Fetch-Dest: webidentity

{
  "provider_urls": [
    "https://accounts.idp.example/config.json"
  ]
}
```

Config endpoint

- no cookies, no origin, no referrer, does not follow redirects beyond eTLD+1
- core endpoints mapping
- branding

```
GET /config.json HTTP/1.1
Host: accounts.idp.example
Accept: application/json
Sec-Fetch-Dest: webidentity

{
  "accounts_endpoint": "/accounts",
  "client_metadata_endpoint": "/client_metadata",
  "id_assertion_endpoint": "/assertion",
  "disconnect_endpoint": "/disconnect",
  "branding": {
    "background_color": "green",
    "color": "#FFEEAA",
    "icons": [{
      "url": "https://idp.example/icon.ico",
      "size": 25
    }],
    "name": "IDP Example"
  }
}
```

Accounts endpoint

- ++ cookies (IdP)
- no origin, no referrer,
does not follow redirects

```
GET /accounts_list HTTP/1.1
```

```
Host: idp.example
```

```
Accept: application/json
```

```
Cookie: 0x23223
```

```
Sec-Fetch-Dest: webidentity
```

```
{  
  "accounts": [  
    {  
      "id": "1234",  
      "given_name": "John",  
      "name": "John Doe",  
      "email": "john_doe@idp.example",  
      "picture": "https://idp.example/profile/123",  
      "approved_clients": ["123", "456", "789"],  
      "login_hints": ["john_doe"],  
      "domain_hints": ["idp.example"],  
    },  
    { ... }  
  ]  
}
```

Client Metadata endpoint

- no cookies
- ++ origin (RP)
- does not follow redirects
- metadata from client registration, such as privacy policy and terms of service URLs

```
GET /client_medata?client_id=1234 HTTP/1.1
Host: idp.example
Origin: https://rp.example/
Accept: application/json
Sec-Fetch-Dest: webidentity

{
  "privacy_policy_url":
    "https://rp.example/cm/privacy_policy.html",
  "terms_of_service_url":
    "https://rp.example/cm/terms_of_service.html"
}
```


Identity Assertion endpoint

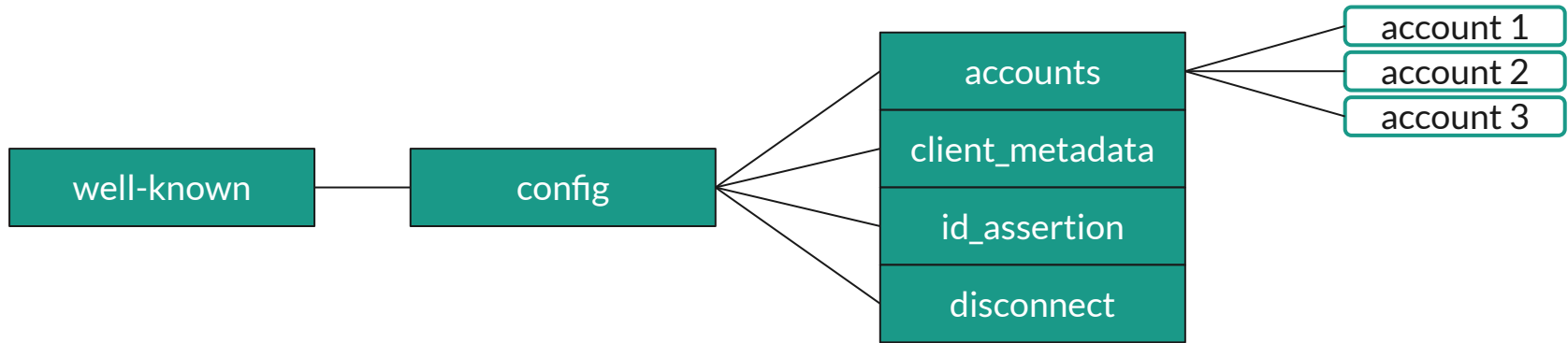
- ++ cookies (IdP)
- ++ origin (RP)
- does not follow redirects
- POST to request an assertion
- Response: opaque string
`{ token }`

```
POST /fedcm_assertion_endpoint HTTP/1.1
Host: idp.example
Origin: https://rp.example/
Content-Type: application/x-www-form-urlencoded
Cookie: 0x23223
Sec-Fetch-Dest: webidentity
```

```
account_id=123
&client_id=client1234
&nonce=Ct60bD
&disclosure_text_shown=true
```

Response

```
{
  "token" : "eyJJC...J9.eyJfQ.SflV_adQssw...5c"
}
```



Relying
Party

User
Agent

Identity
Provider

```
navigator.credentials.get({  
  identity: {  
    providers: [{  
      configURL: ...  
    }]  
  }  
})
```

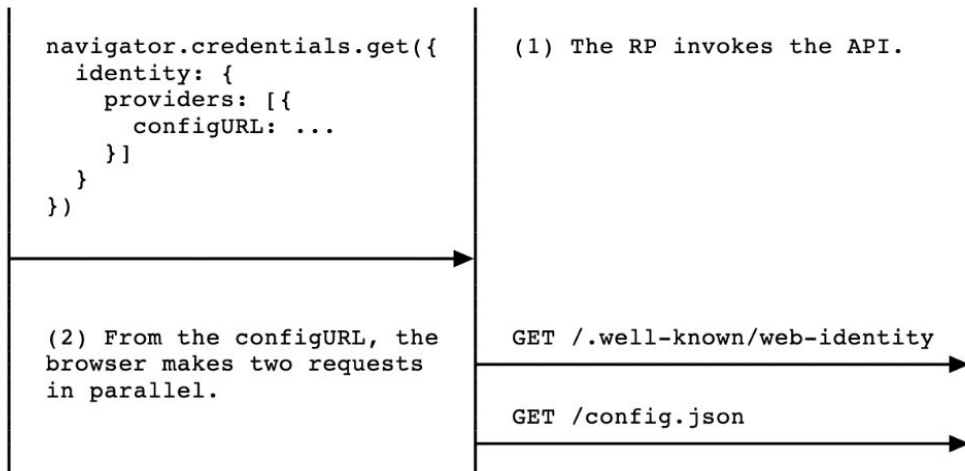
(1) The RP invokes the API.



Relying
Party

User
Agent

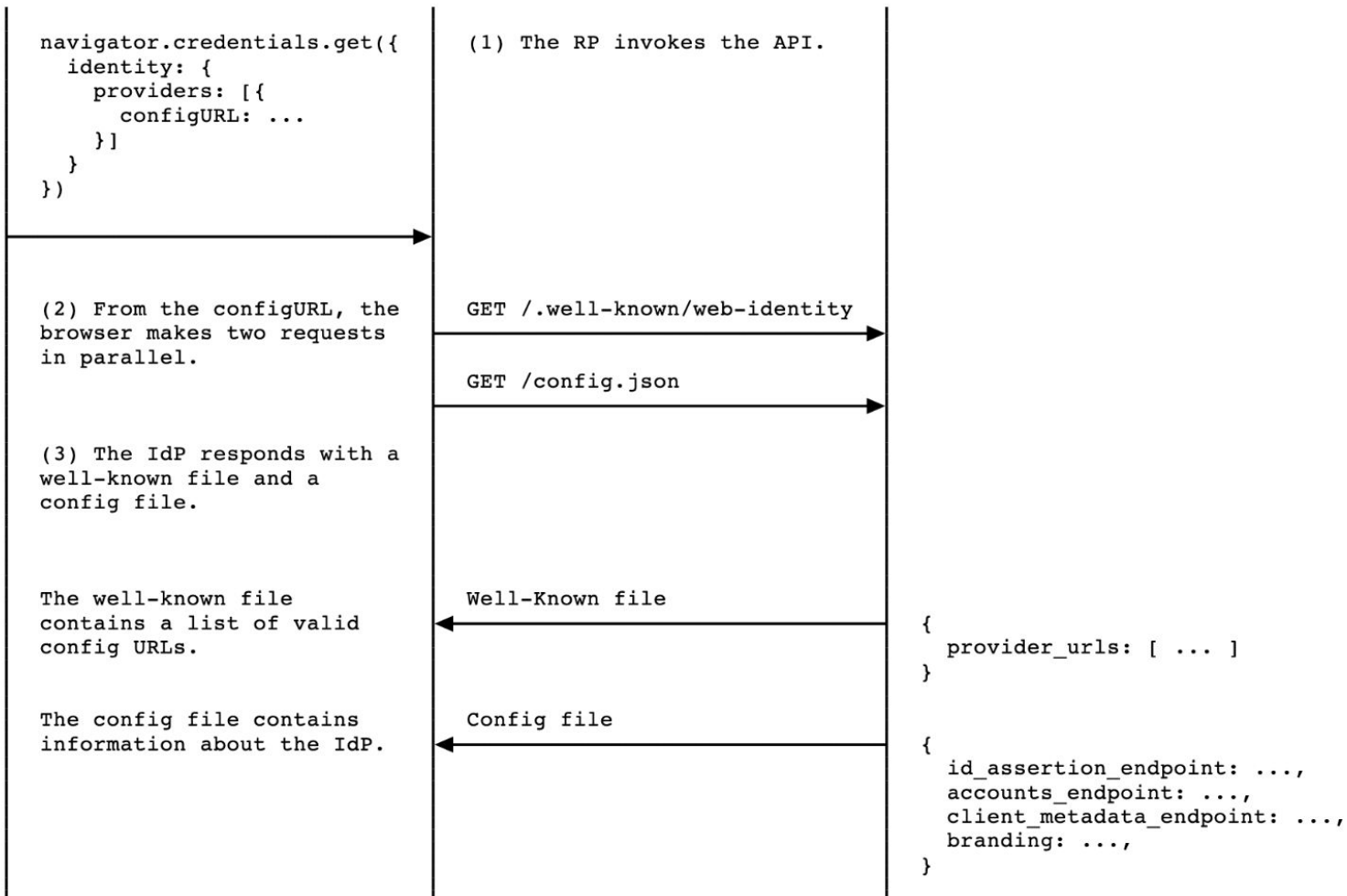
Identity
Provider



Relying
Party

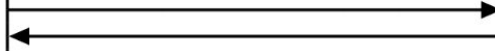
User
Agent

Identity
Provider



(4) The browser proceeds to fetch the list of accounts that the user is logged in to.

GET accounts.php
cookies



```
{  
  accounts: [ ... ]  
}
```

(4) The browser proceeds to fetch the list of accounts that the user is logged in to.

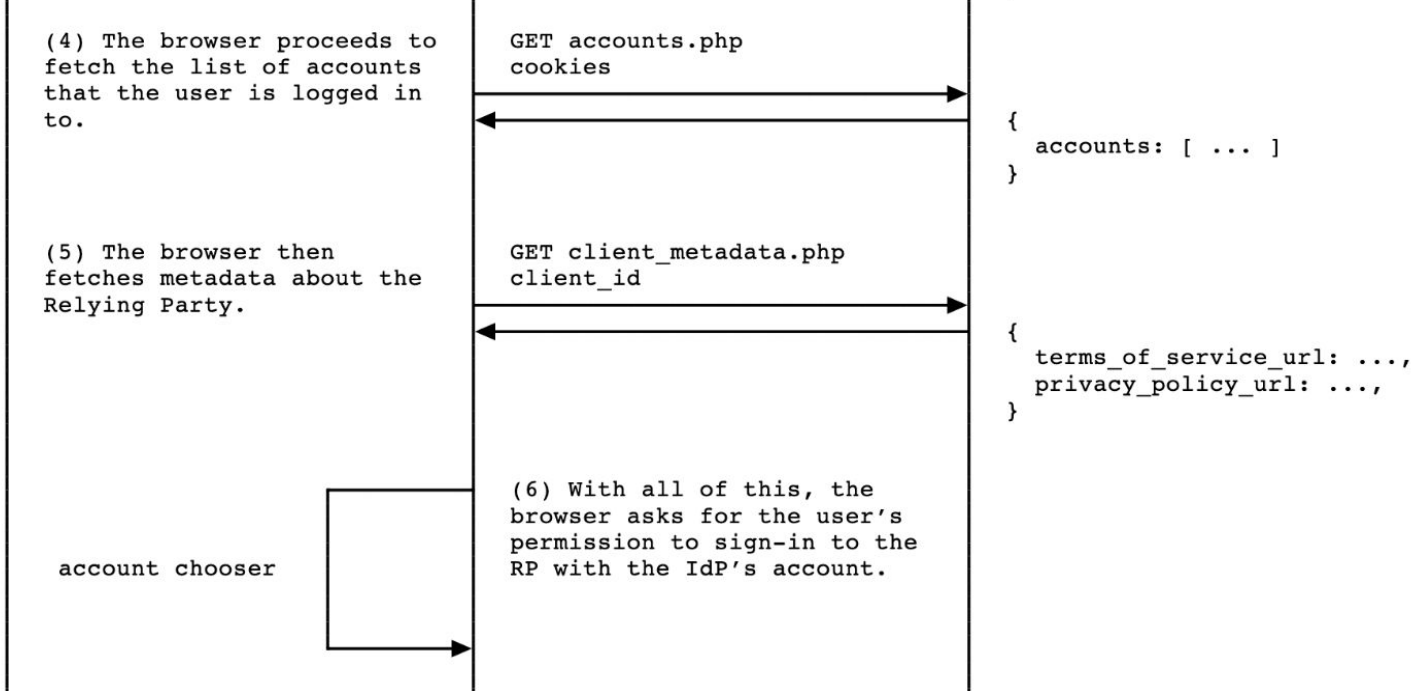
GET accounts.php
cookies

```
{  
  accounts: [ ... ]  
}
```

(5) The browser then fetches metadata about the Relying Party.

GET client_metadata.php
client_id

```
{  
  terms_of_service_url: ...,  
  privacy_policy_url: ...,  
}
```



(4) The browser proceeds to fetch the list of accounts that the user is logged in to.

GET accounts.php
cookies

```
{  
  accounts: [ ... ]  
}
```

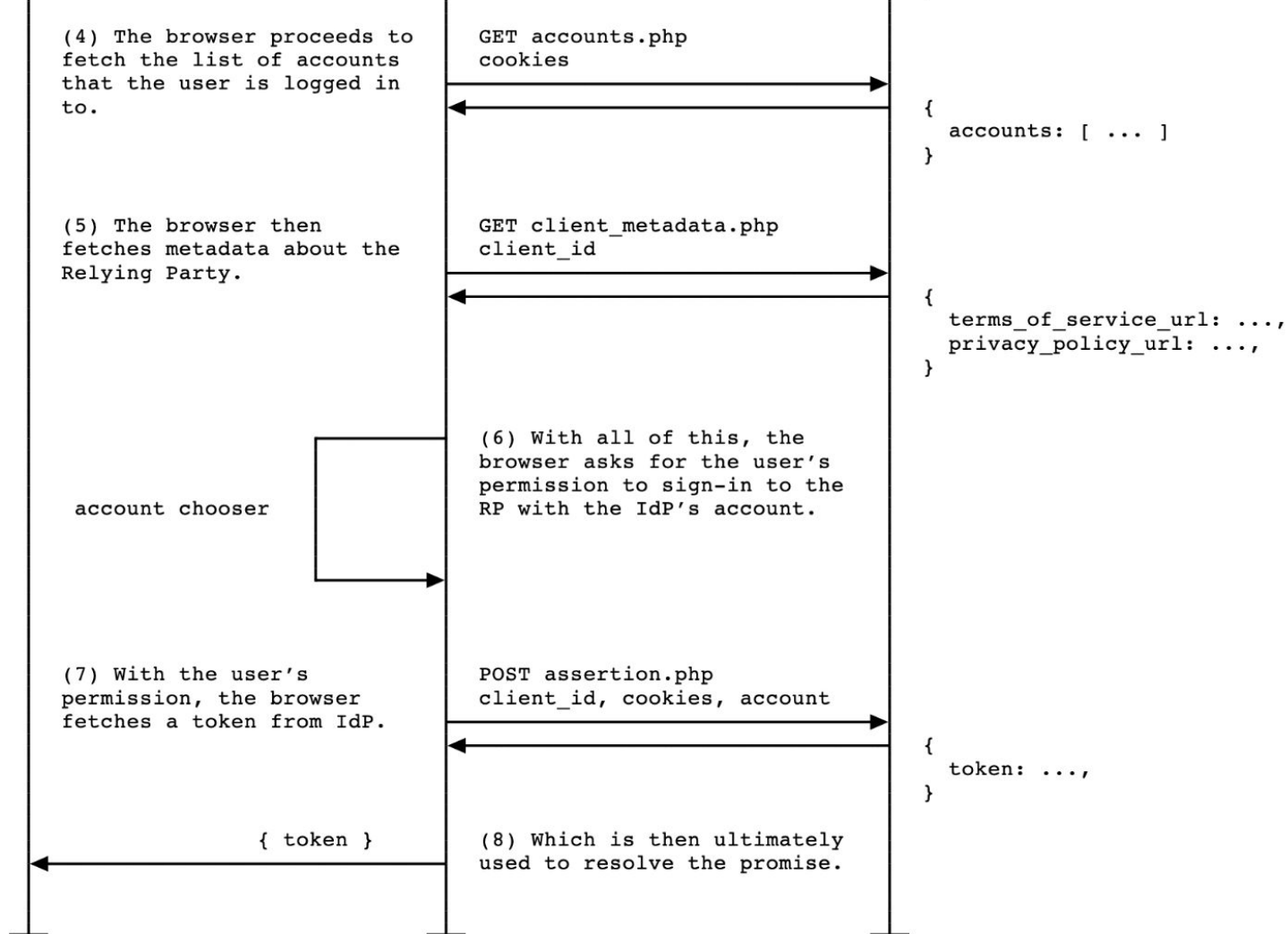
(5) The browser then fetches metadata about the Relying Party.

GET client_metadata.php
client_id

```
{  
  terms_of_service_url: ...,  
  privacy_policy_url: ...,  
  ...  
}
```

account chooser

(6) With all of this, the browser asks for the user's permission to sign-in to the RP with the IdP's account.



Disconnect endpoint

- ++ cookies (IdP)
- ++ origin (RP)
- does not follow redirects
- POST to disconnect endpoint
- RP calls:

```
IdentityCredential.disconnect({  
  configURL: "https://idp.com/config.json",  
  clientId: "rp123",  
  accountHint: "account456"  
});
```

```
POST /fedcm_disconnect_endpoint HTTP/1.1  
Host: idp.example  
Origin: https://rp.example/  
Content-Type: application/x-www-form-urlencoded  
Cookie: 0x23223  
Sec-Fetch-Dest: webidentity  
  
client_id=client1234  
&account_hint=hint12
```



Demo

Status

Spec Status

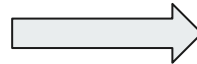


W3C
Federated Identity
Community Group

new incubations & ideas

community discussion & engagement

incubation & early spec work



W3C
Federated Identity
Working Group

formed March 2024

expected to kickoff in June 2024

W3C Recommendation Track

Implementation Status



Shipped in
Chrome 108



Shipped in
Edge



Feature Flag ¹



Supportive ²

[1] `dom.security.credentialmanagement.identity.enabled`

[2] `tcslices.link/osw-fedcmwebkit`

Collaboration w/ OpenID and OAuth WGs



- What would an OpenID Connect binding for FedCM profile look like?
- What are the intersections with OpenID Federation and OpenID Discovery?

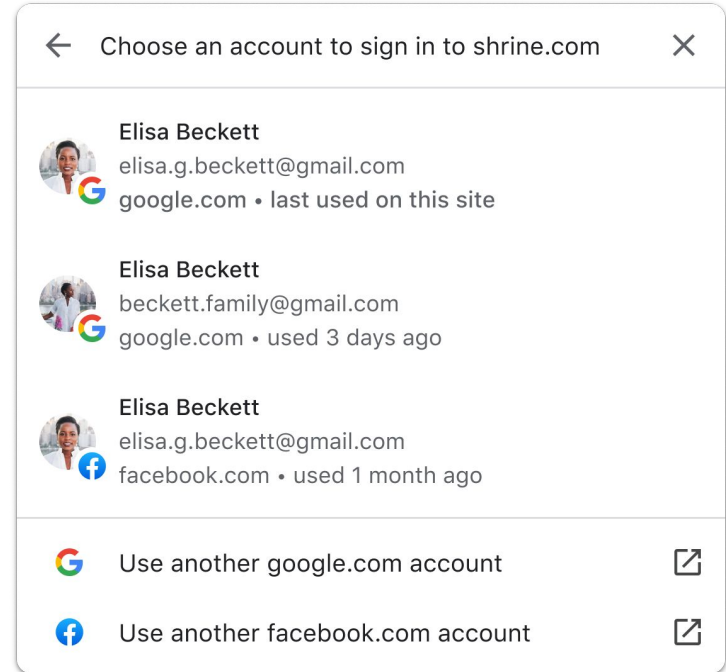
What's next?

Multiple IdP Support

challenging with multiple identity SDKs all calling FedCM at the same time

- when to show the selection UI?
- which order should providers be in as they load?
- how should providers "pop" into the list as they load?

first experiment is to allow multiple providers in a single get call (e.g. non-SDK or one SDK)



IdP Registration (DevTrial)



USE CASE

RP will accept IdPs they're not explicitly aware of (for example, smaller IdPs or large federations)

HOW IT WORKS

The IdP calls a registration API during user sign in

The RP includes a provider with a configURL of "any"

This is an early implementation that is soliciting feedback!

```
navigator.credentials.get({
  identity: {
    providers: [{
      configURL: "any",
      clientId: "https://rp.example",
      nonce: "123",
    }]
  }
});
```

```
IdentityProvider.register('https://idp.example/fedcm.json');
```

The image shows a Chrome browser window with the following elements:

- Address Bar:** Shows "google.com" with navigation icons (back, forward, refresh) and a search icon.
- Permissions Dialog:** A dark overlay box with the text "www.google.com wants to Use your accounts to login to websites" and two buttons: "Block" and "Allow".
- Page Content:** The Google logo is displayed with a festive string of colorful lights.
- Developer Tools Console:** Opened on the right, showing a red error message: "[Violation] Permissions policy m=cdo... violation: unload is not allowed in this document." Below the error, it says "Autofocus processing was blocked because a document already has a focused element." and shows a code snippet: `> await IdentityProvider.register("https://www.google.com/fedcm.json")`.

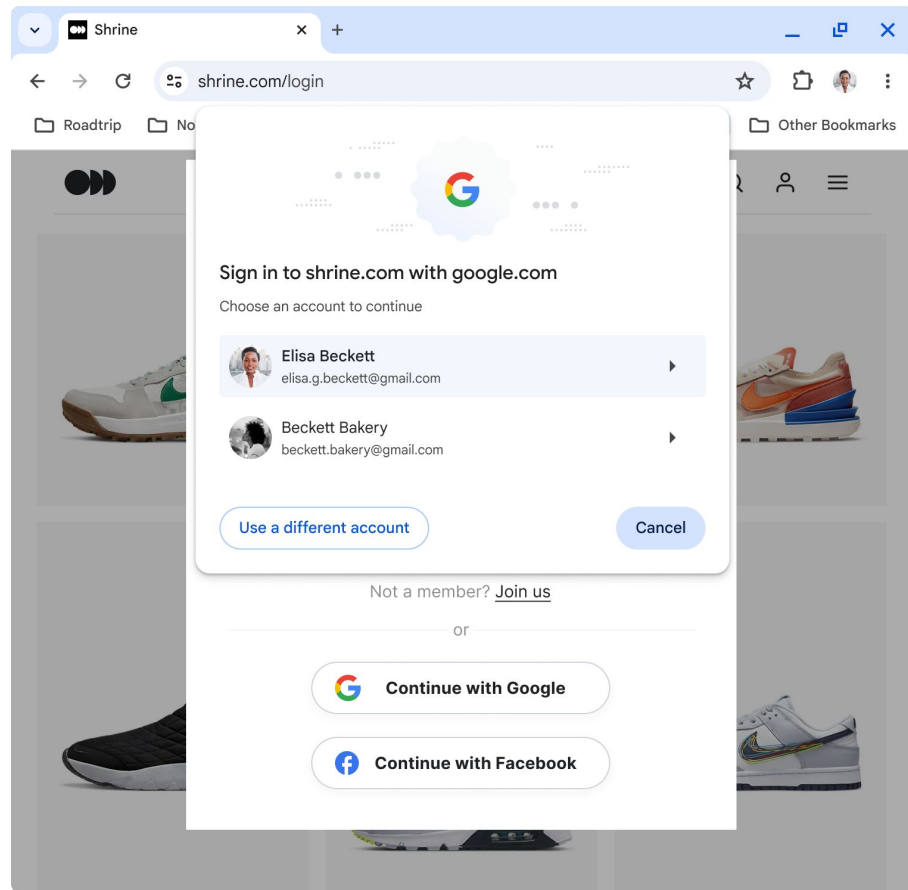
Button Mode

USE CASE

Need to facilitate a sign in when the user doesn't yet have a session with the IdP

DETAILS

- Modal Dialogs
- Graceful degradation when users are signed-out of the IdP
- Use Another account affordance
- Coming to Chrome in Origin Trials in M125



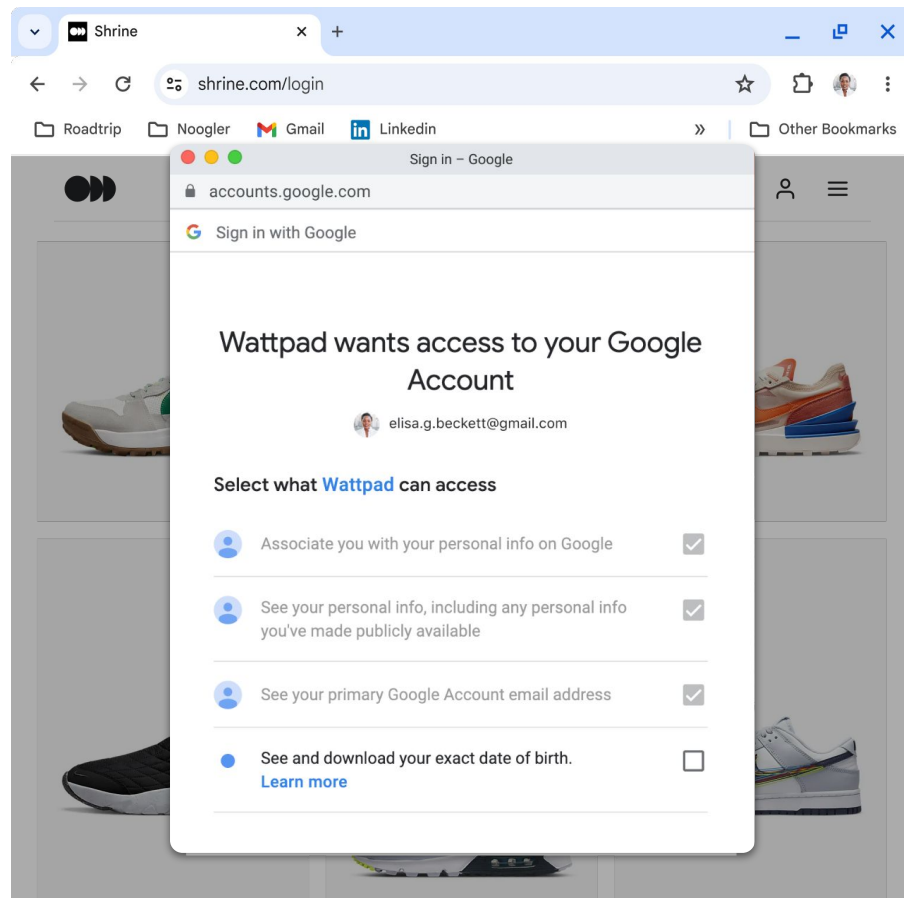
Continuation API

USE CASE

User needs to perform an additional action prior to continuing (e.g. consent, step up authentication, etc)

DETAILS

- Pop-up window that the IdP can use to gather further permission from the user
- Coming to Chrome in Origin Trials in M126



FedCM with Storage Access API



USE CASE

Scenarios that still required unpartitioned cookies for things like token refresh or embedded content

HOW IT WORKS

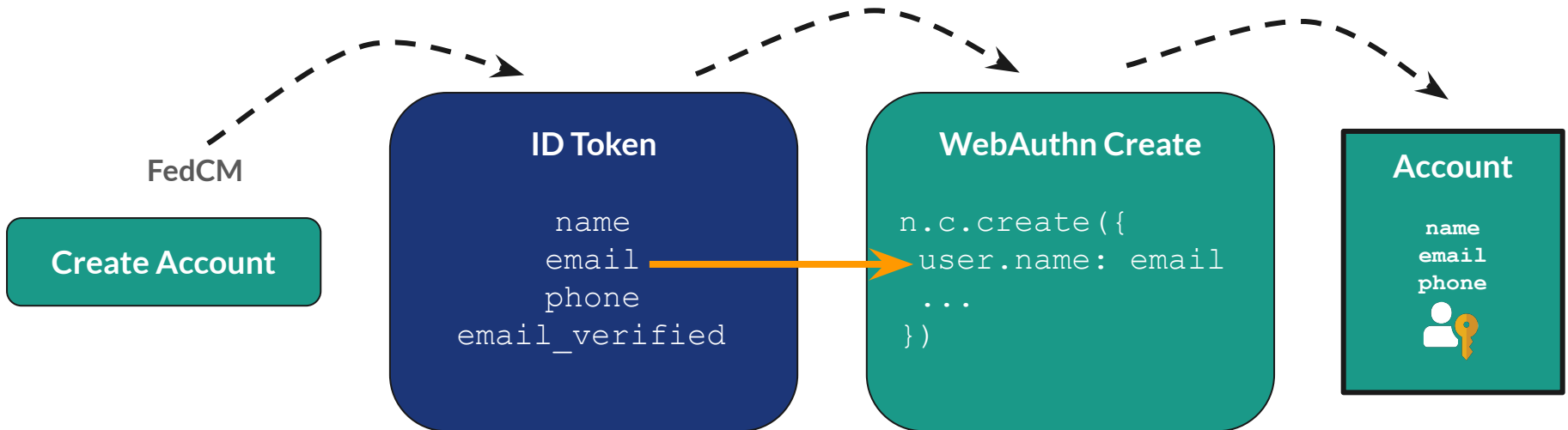
- User signs in using a federated identity provider via FedCM API
- Storage Access API called from iframe to access content or request a token
- Browser checks for connected account in internal database
- Storage Access granted without a prompt

Intersection with Passkeys



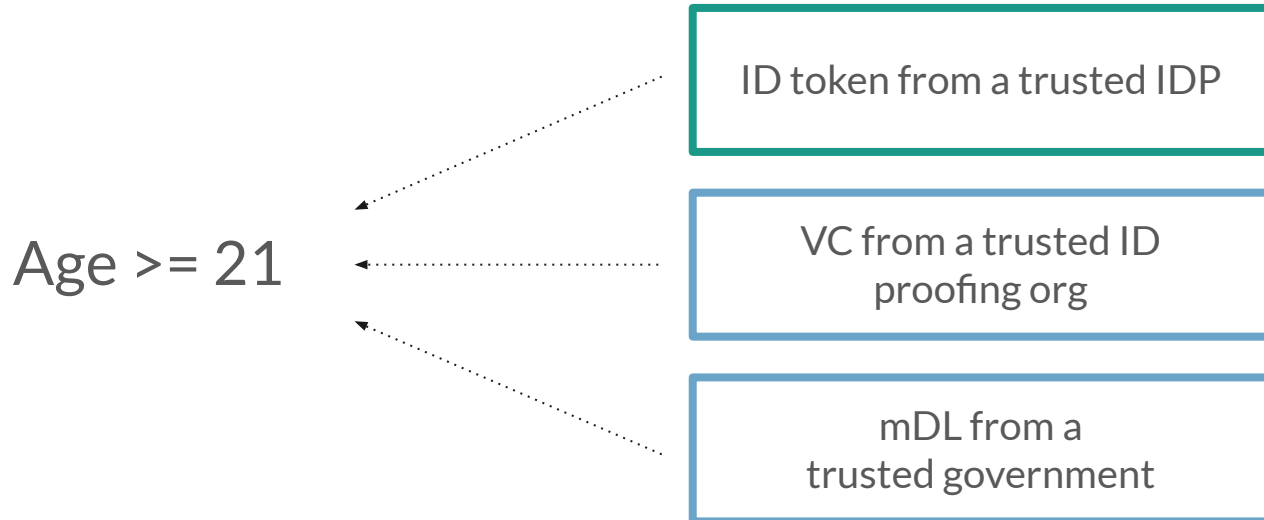
federation for *sign up*

passkeys for *sign in* 



Intersection with Digital Credentials

There's quite a bit of intersection between **federation** and **digital credentials** (e.g. verifiable credentials)



Intersection with Passkeys and Digital Credentials



federation for sign up

passkeys for sign in

digital credentials for
additional claims

Discussion!

Thanks!