# Proposed changes to First-Party Sets

Kaustubha Govind & Johann Hofmann

# Agenda

- Background: Feedback on FPS
- Proposal Overview
- Proposal Details
- Pause for Q&A
- Key Questions for Feedback
- Next call and meeting cadence

# Feedback on First-Party Sets

## Opposing

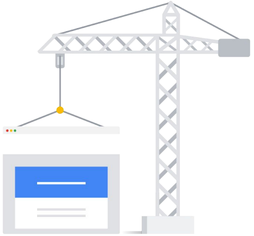
Apple, Mozilla and others expressed various concerns with FPS. Points of discussion were cookie access through SameParty, policy concerns, and user understanding.

## Supportive

Besides Edge & Chrome, some web developers & businesses have voiced support for the proposal. They signalled concerns about cross-browser interoperability, among other things.

# Goals for proposed changes

- Continued focus on **addressing user-facing breakage**
- Index on user expectations and experience rather than corporate structure
- Better understanding and separation of use cases
- Allow User Agents to mediate and restrict cookie access within sets
- Interoperability with browsers that do not support FPS



Google

# A New Proposal

In three parts

## Abandon SameParty

SameParty cookies enable synchronous cross-site linking within First-Party Sets using a new cookie attribute.

This is very easy to implement for developers, but stands in the way of better interoperability and prevents browsers from mediating or intervening based on async choices.

## Adopt the Storage Access API & Improve it

Without SameParty cookies, the SAA is the new mandatory path to gain cross-site cookie access.

Browsers wishing to integrate with FPS (such as Chrome) could **automatically grant SAA requests**.

We'll work on improving the SAA, starting with the **rSAForSite** proposal.

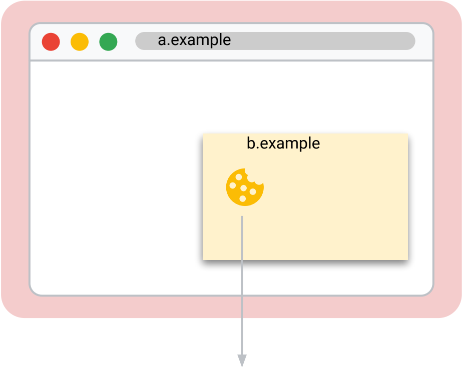
## Introduce Subsets

Subsets can help us better understand how multi-domain sites are structured and what use cases are being supported. Different browser handling policies and criteria can be applied depending on the use case.

# SameParty vs. SAA

Where a.example and b.example are members of the same set.

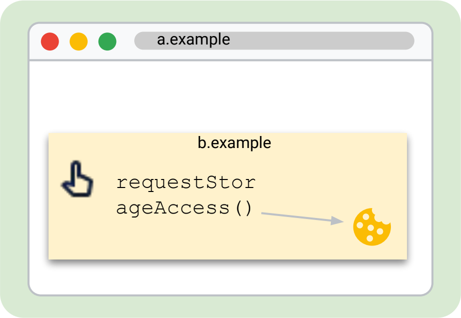
## SameParty



b.example can access cross-site cookie set with:

```
Set-Cookie: cookie=tasty;
SameSite=None; Secure; SameParty
```

## requestStorageAccess



1. With user interaction, b.example calls rSA, which runs custom browser steps (such as a prompt) to determine access.

**FPS-supporting browsers could auto-grant for certain set members** instead of prompting.

2. If granted, b.example can access cross-site cookie set with:

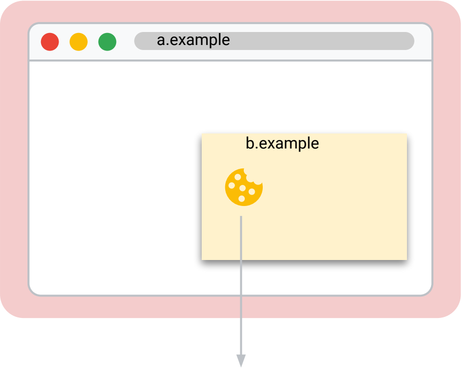
```
Set-Cookie: cookie=tasty;
SameSite=None; Secure;
```

b.example needs to notify a.example if it requires refreshing other resources on the page

# SameParty vs. SAA + rSAForSite

Where a.example and b.example are members of the same set.

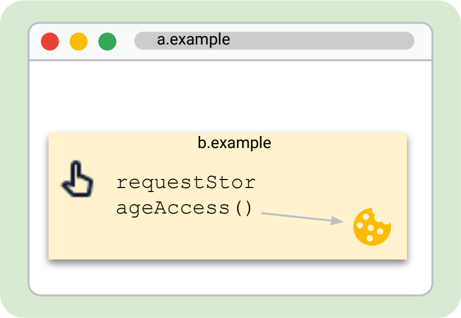
## SameParty



b.example can access cross-site cookie set with:

```
Set-Cookie: cookie=tasty;
SameSite=None; Secure; SameParty
```

## requestStorageAccess

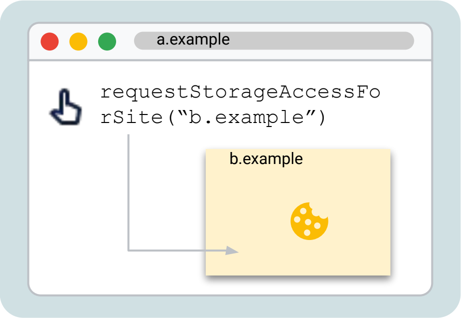


```
requestStorageAccess()
```

1. With user interaction, b.example calls rSA, which runs custom browser steps (such as a prompt) to determine access.

FPS-suppporting browsers could auto-grant for certain set members instead of prompting.

## ...forSite (Proposed)



```
requestStorageAccessForSite("b.example")
```

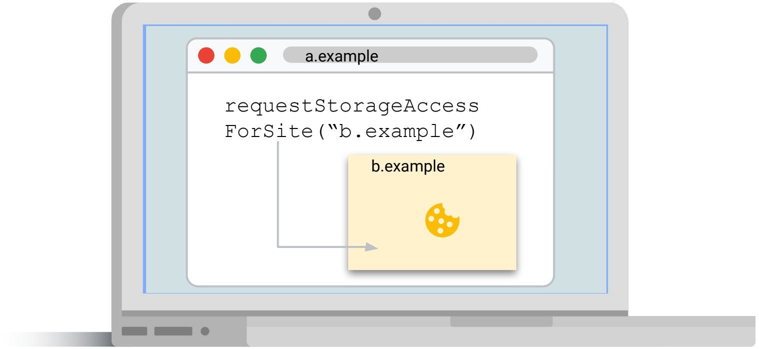
1. With user interaction, a.example calls rsaForSite *on behalf* of b.example, which runs custom browser steps to determine access.

2. If granted, b.example can access cross-site cookies.

# requestStorageAccessForSite Proposal

Main considerations:

- Provide an easier integration for developers without iframe-based cross-site resource flows.
- Non-iframe solutions were previously suggested in [#53](#53) and [#83](#83)
- There's prior art from both Safari and Firefox implementing this as a helper function for quirks/shims only accessible to browser internals.
- Likely needs to be gated on a strong browser trust mechanism (such as, but not limited to, FPS).



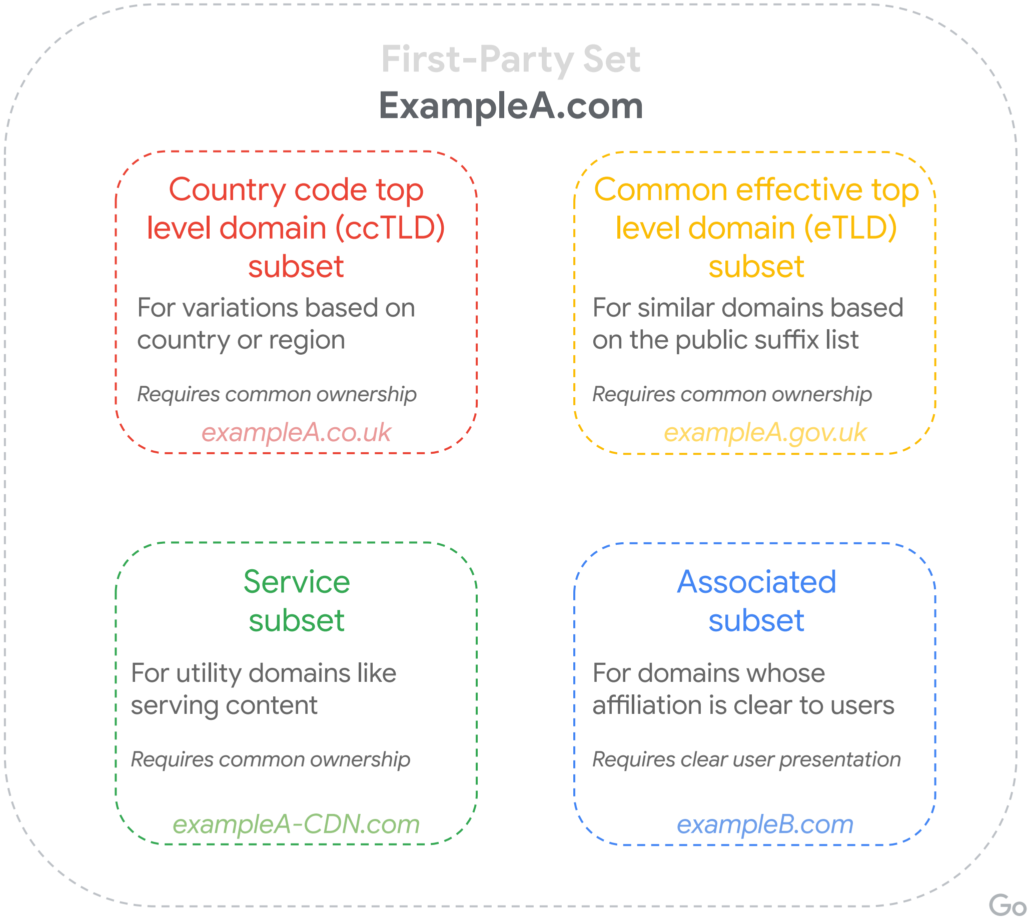
```
requestStorageAccess
ForSite("b.example")
```

[https://github.com/mreichhoff/requestStorageAccessForSite](https://github.com/mreichhoff/requestStorageAccessForSite)

# Subsets

The new framework defines different types of subsets that set primaries can declare at the time of submission.

Instead of using a single definition to apply to a range of use cases, we propose granular criteria and handling to be applied by use case by specifying "subsets."



First-Party Set

**ExampleA.com**

Country code top level domain (ccTLD) subset

For variations based on country or region

*Requires common ownership*

*exampleA.co.uk*

Common effective top level domain (eTLD) subset

For similar domains based on the public suffix list

*Requires common ownership*

*exampleA.gov.uk*

Service subset

For utility domains like serving content

*Requires common ownership*

*exampleA-CDN.com*

Associated subset

For domains whose affiliation is clear to users

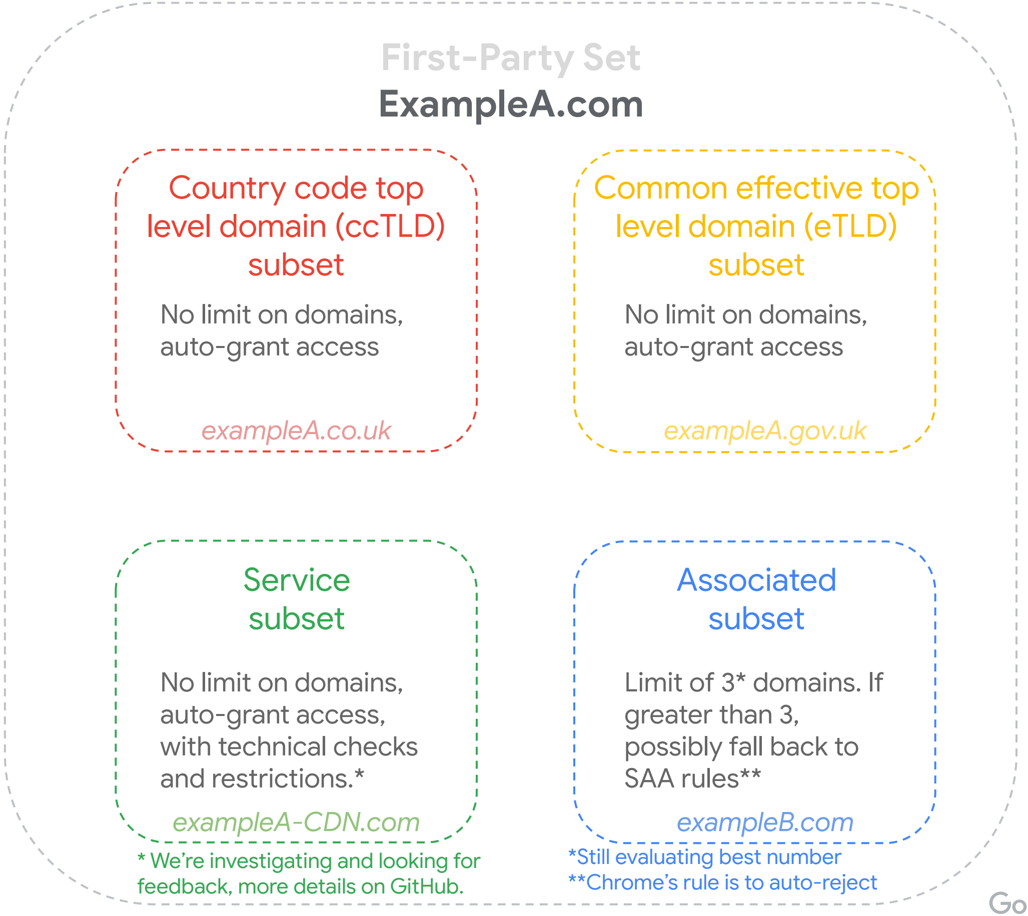
*Requires clear user presentation*

*exampleB.com*

# Per-subset criteria

While the subset framework can create transparency for why a domain has been added to a set, the primary value to this framework is that the browser could handle each subset differently.

This increased flexibility is an improvement over the previous FPS policy, which required the same requirements for every domain in set, regardless of the type of domain that it was.



First-Party Set

ExampleA.com

Country code top level domain (ccTLD) subset

No limit on domains, auto-grant access

exampleA.co.uk

Common effective top level domain (eTLD) subset

No limit on domains, auto-grant access

exampleA.gov.uk

Service subset

No limit on domains, auto-grant access, with technical checks and restrictions.*

exampleA-CDN.com

Associated subset

Limit of 3* domains. If greater than 3, possibly fall back to SAA rules**

exampleB.com

* We're investigating and looking for feedback, more details on GitHub.

*Still evaluating best number

**Chrome's rule is to auto-reject

# What does this mean for developers?

Submit set to public tracker

- Set manifest
- Demonstrate domain control via .well-known files
- Must pass subset-specific technical checks
- Public assertion about conformance to requirements

Sample set manifest:

```
{
  "ccTLDs": {
    "exampleA.com": ["exampleA.ca", "exampleA.co.uk"],
    "exampleB.com": ["exampleB.co.uk"],
    // etc.
  },
  "primary": "exampleA.com",
  "associatedDomains": ["exampleB.com", "exampleC.com"],
  "serviceDomains": ["exampleA-usercontent.com"],
}
```

Client side code on exampleA.com, using...

**rSA:**

```
<iframe src='https://exampleB.com/example.html'>
 // code on exampleB.com/example.html
 button.addEventListener('click', function(){
  document.requestStorageAccess()
      .then(/*integrate with exampleA.com*/)
      .catch(/*handle potential errors
 });
</iframe>
```

or

**rSAForSite:**

```
button.addEventListener('click', function(){
 document.requestStorageAccessForSite('https://exampleB.com')
     .then(/*add exampleB.com resources now*/)
     .catch(/*handle potential errors*/);
});
```

# Key Questions for Feedback

**Subsets** - What are the right subsets to choose initially? Are there use cases that don't work with subsets anymore? (GitHub issue #96)

**Associated** - What challenges might developers encounter from a limit of 3 associated domains? (GitHub issue #93)

**CHIPS** - How does CHIPS interact with (and benefit from) FPS and SAA? (GitHub issue #94)

**rSAForSite** - Can we solve existing use cases for FPS with embedded rSA? If not, does the rSAForSite proposal help? (GitHub issue #97)

**Fraud** - What additional technical checks should we consider to combat abuse? (GitHub issue #95)