IETF 119
Brisbane
March 2024

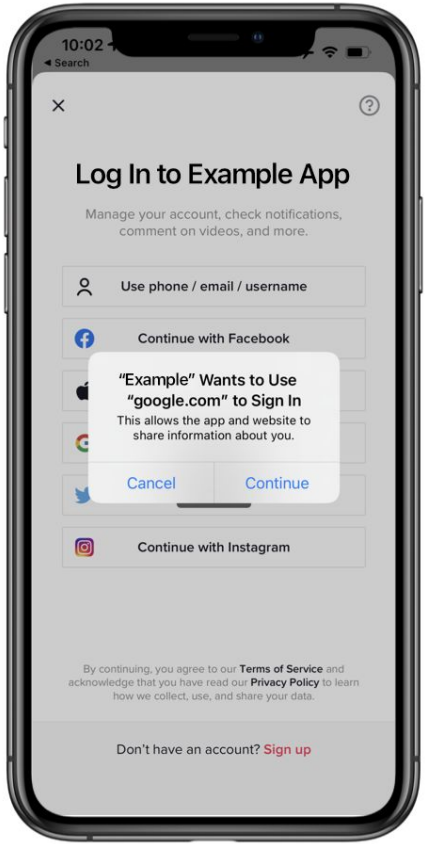
Aaron Parecki
George Fletcher
Pieter Kasselman

OAuth for First-Party Apps

https://datatracker.ietf.org/doc/draft-parecki-oauth-first-party-apps/
draft -01

Why?

Developers want a better user experience for first-party apps



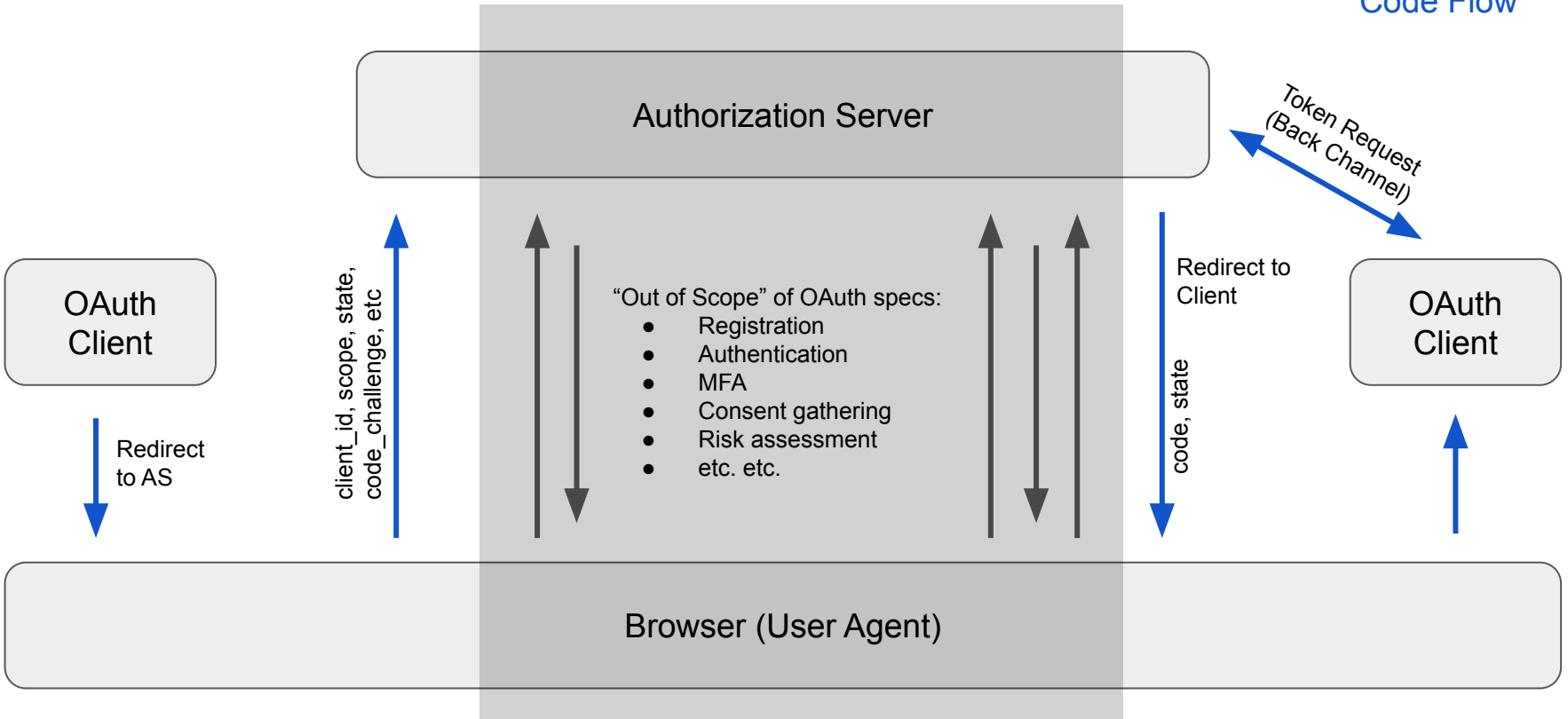
What is happening today

People are finding workarounds to avoid RFC8252

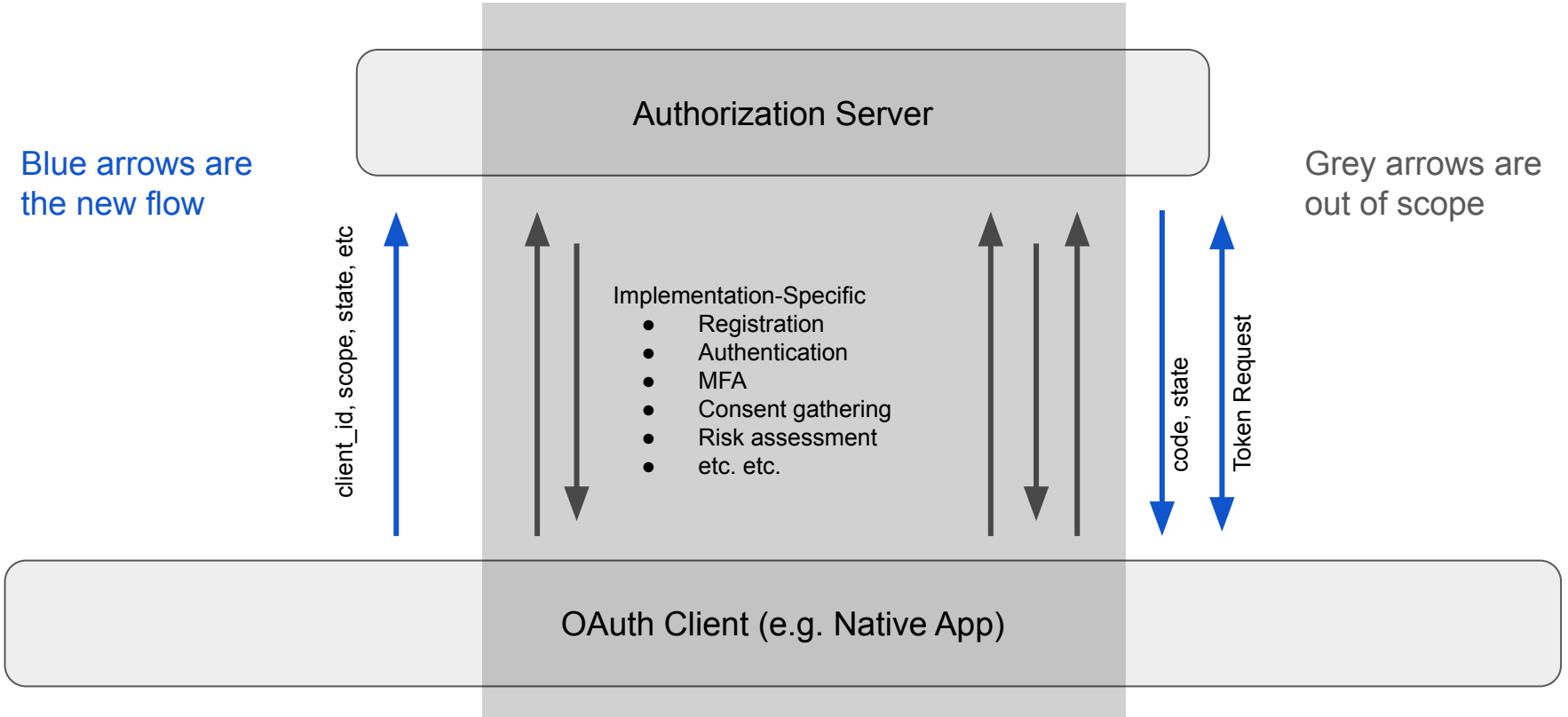
- Custom DIY solutions for native apps
- Using Resource Owner Password Grant
  - (Unable to add MFA)
- OAuth servers creating proprietary APIs to facilitate direct interaction with native apps
- Scripting hidden web views to emulate user interaction with the AS

Authorization Code Flow for Web Apps

Blue arrows are the OAuth Authorization Code Flow



First-Party Apps Flow



Goals

- Reuse existing OAuth building blocks as much as possible
- Mirror the web authorization code flow, defining how the client starts and ends the flow
  - Leave the specifics of the user authentication out of the core framework
- Specifics of user authentication can be proprietary to an AS as they are today, or can be defined as extensions
  - Especially if based on standards like FIDO

Authorization Challenge Endpoint

- New endpoint
- Accepts parameters that would have been included in the query string to the authorization endpoint
  - including any extensions such as Resource Indicators, OpenID Connect, JAR, etc
- Accepts POST from client to start and continue an authorization
  - The AS defines what the client sends in the requests and defines its own error responses
- Response is an authorization code, error, or redirect
  - The AS may want to interact with the user directly, e.g. based on risk assessment, new authentication method not implemented in the app, or exceptions like account recovery

Token Endpoint

- No changes to the token request
- Client POSTs the directly-obtained authorization code to get an access token

Authorization Challenge Endpoint

Why a new endpoint?

- Existing authorization endpoint is never interacted with by the OAuth client today, only by the browser
- It expects to receive requests from a **User Agent**, and return **HTML**
- Feedback has indicated people are unwilling to modify their existing authorization endpoint to accept a direct POST from a client and return JSON

Error Responses

Token Endpoint Error Response

- Any request to the token endpoint (e.g. with refresh token) can fail with an error indicating the client needs to obtain a fresh authorization from the user and start the flow over

Resource Server Error Response

- No changes made by this draft
- Can use Step-Up Authentication to tell the client to start a new flow

Next: To be determined…

Changes since IETF 118

- Updated authorization code binding with DPoP (#59)
- Removed "`ash`" claim for DPoP binding (#58)
- Enable PAR as an optional optimization for "redirect to web" (#46)
  - Turned "redirect to web" response into an error response
  - Also added "`request_uri`" to error response
  - If the client expects "redirect to web" frequently, it can include a PKCE code challenge in the initial authorization challenge request
- Clarified follow-up requests are not required to be form-encoded (#50)
  - Feedback from Brian Campbell at IETF 118
- Require `auth_session` is bound to a device (#57)
  - Prohibits moving sessions across devices

Generic OAuth POST Request Framework?

Sent email to the list on Feb 2, 2024, but did not see any interest

https://mailarchive.ietf.org/arch/msg/oauth/Q8O7ITOyAsx_wlgVctsItcheaNM/

No plans to pursue this further at this time

Next Steps

- Working group adoption?
- Create an extension to this draft for passkeys