



OAuth 2.1

The Future of API Security

Rob Allen, October 2025



OAuth is the standard for securing access to APIs



OAuth 2.0

A Refresher



OAuth 2.0 Roles

- Resource Owner (The User)
- Resource Server (The API)
- Client (The application that uses the API)
- Authorization Server (OAuth server)



OAuth 2.0 Protocol Flows

- Resource Owner Password Credentials
- Authorization Code
- Implicit
- Client Credentials

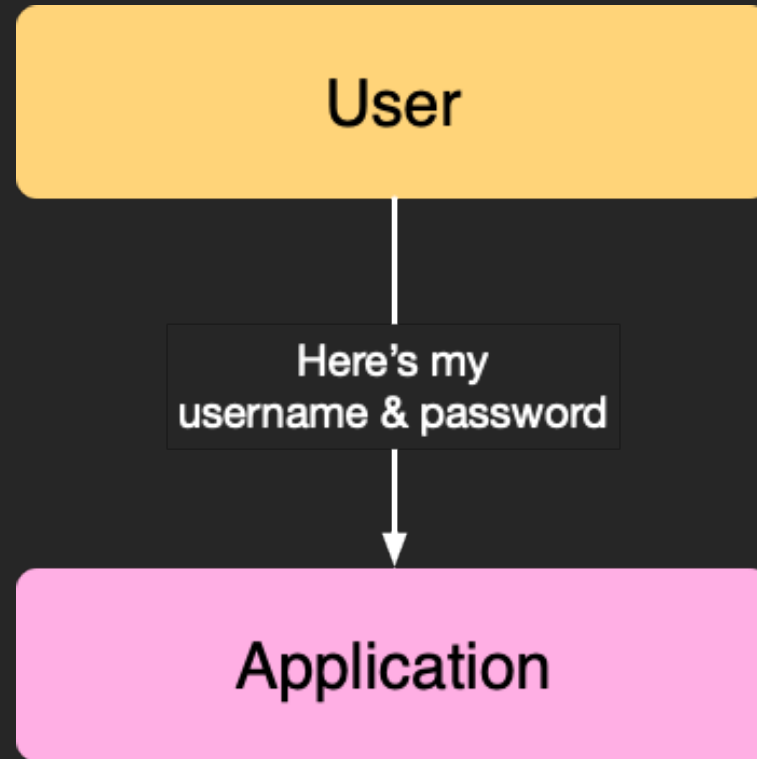


Password Credentials Flow

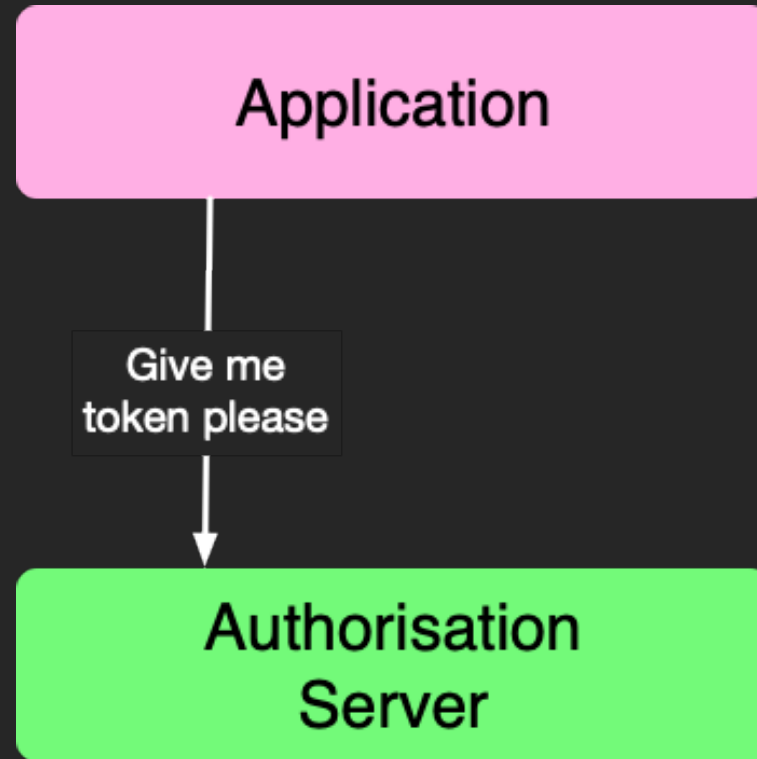
For logging into 1st party apps



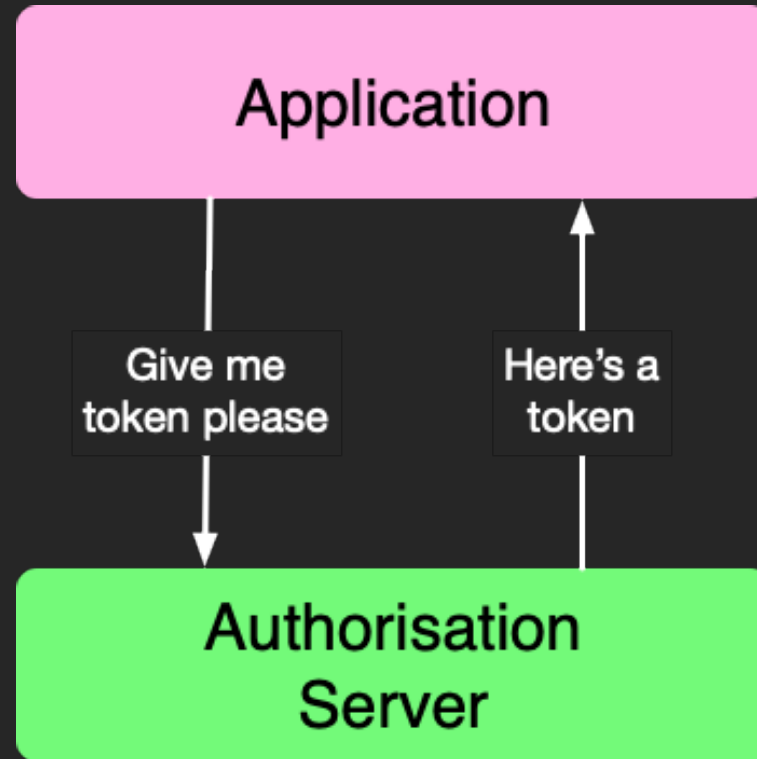
Password Credentials Flow



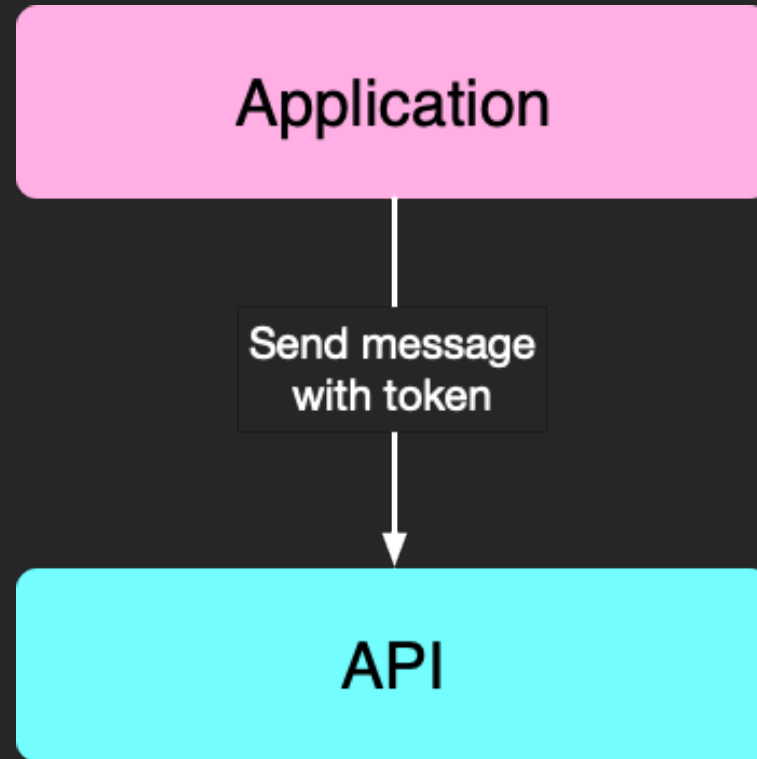
Password Credentials Flow



Password Credentials Flow



Password Credentials Flow

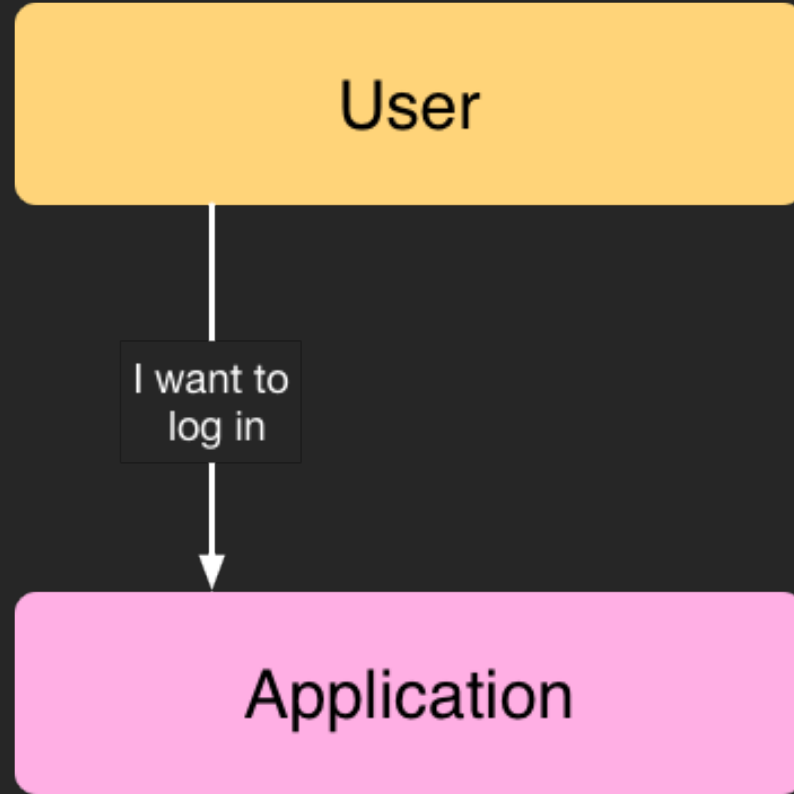


Authorization Code Flow

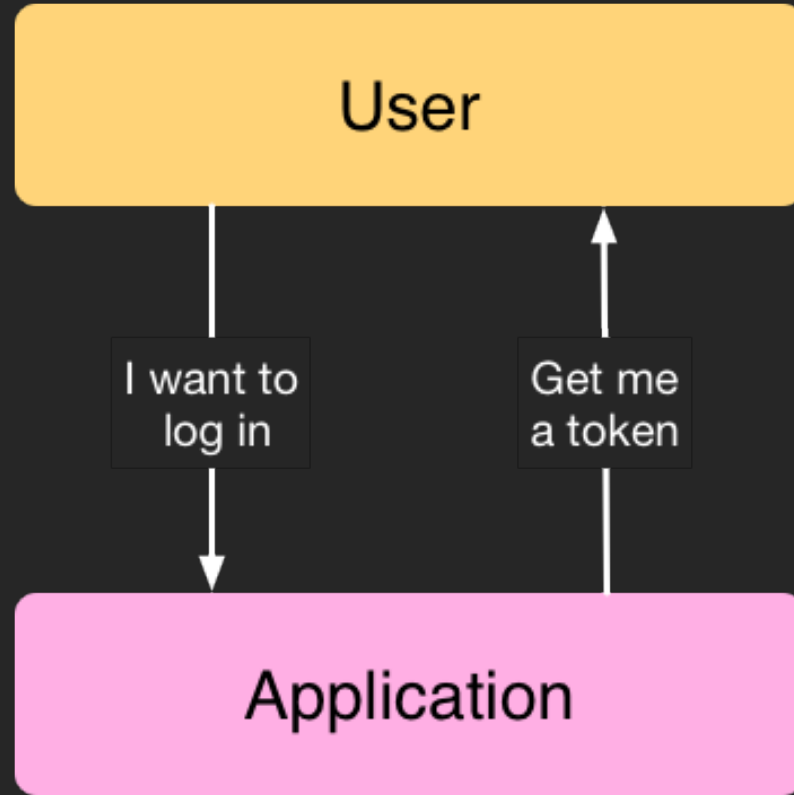
For logging into 3rd party websites



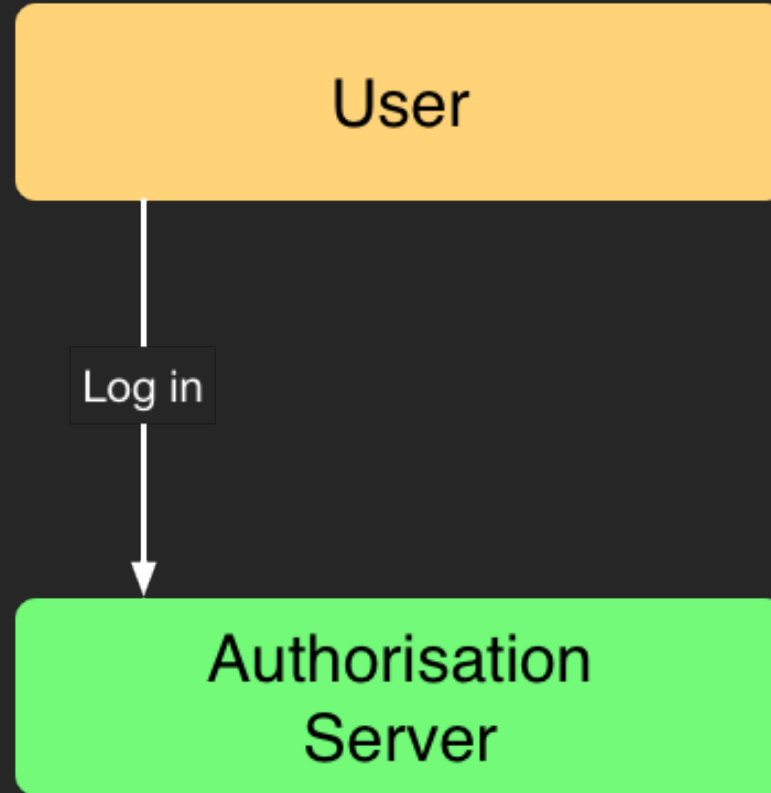
Authorization Code Flow



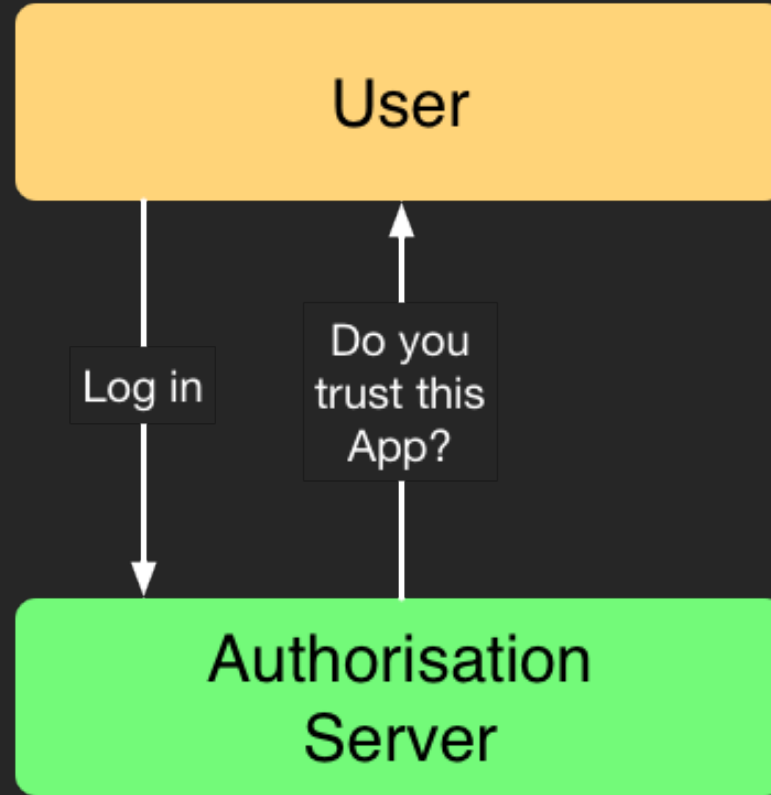
Authorization Code Flow



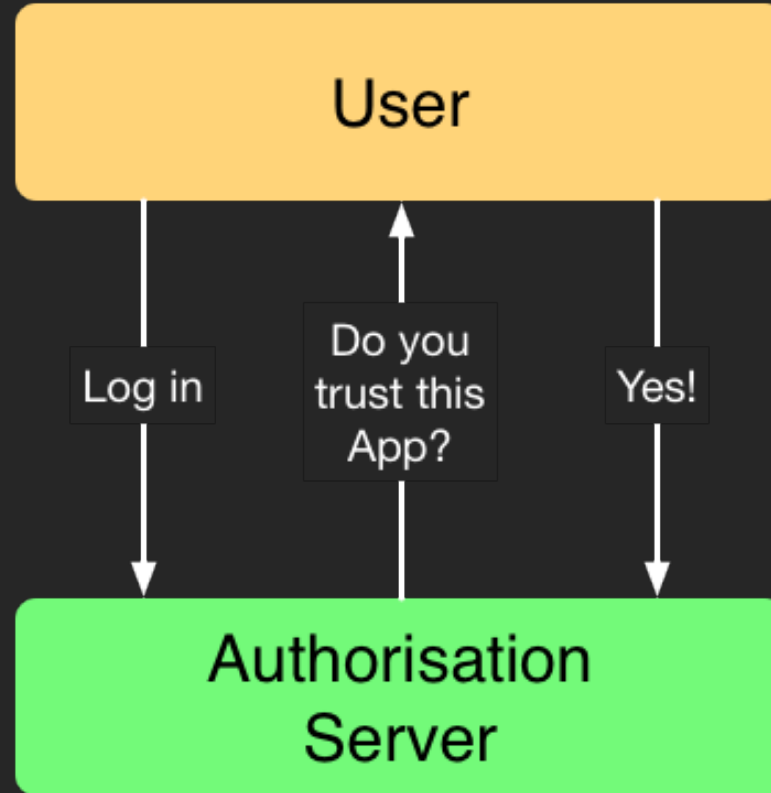
Authorization Code Flow



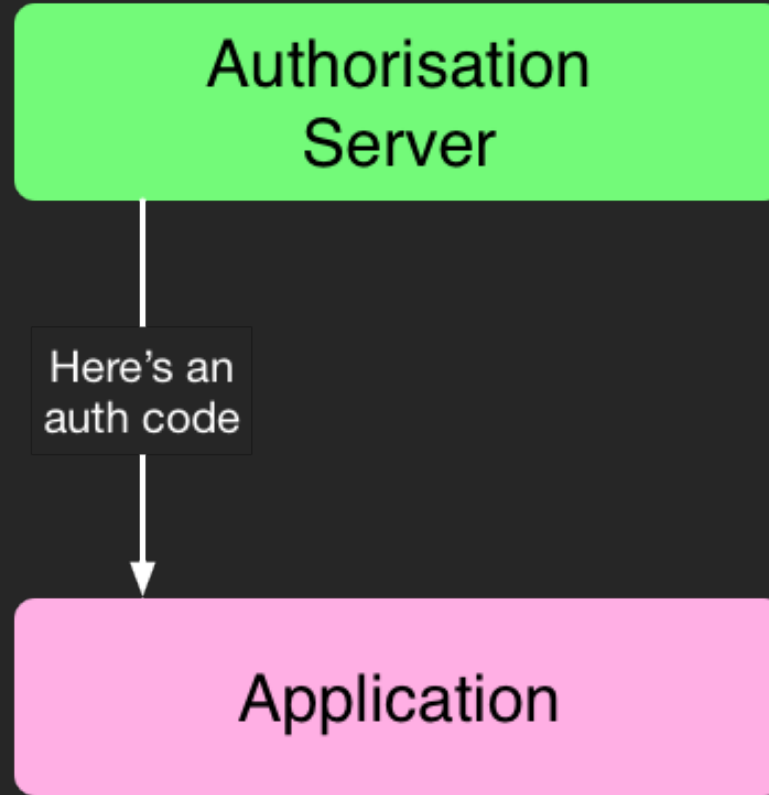
Authorization Code Flow



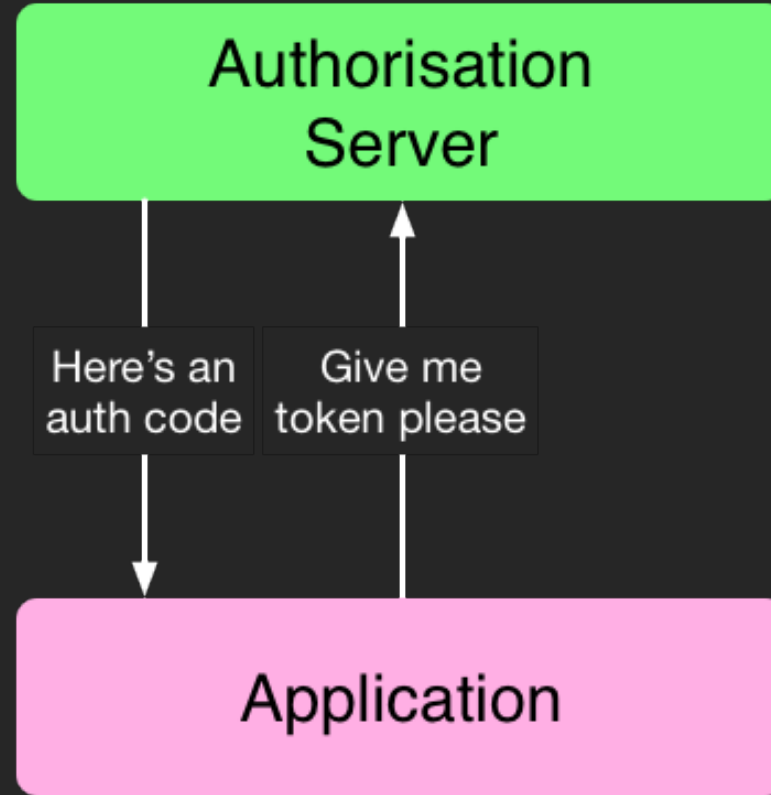
Authorization Code Flow



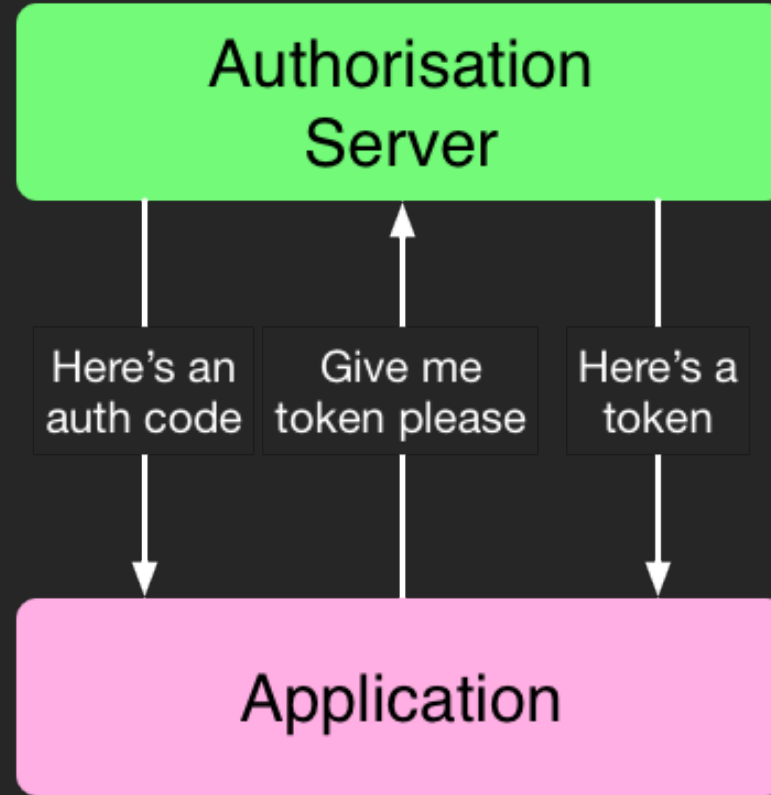
Authorization Code Flow



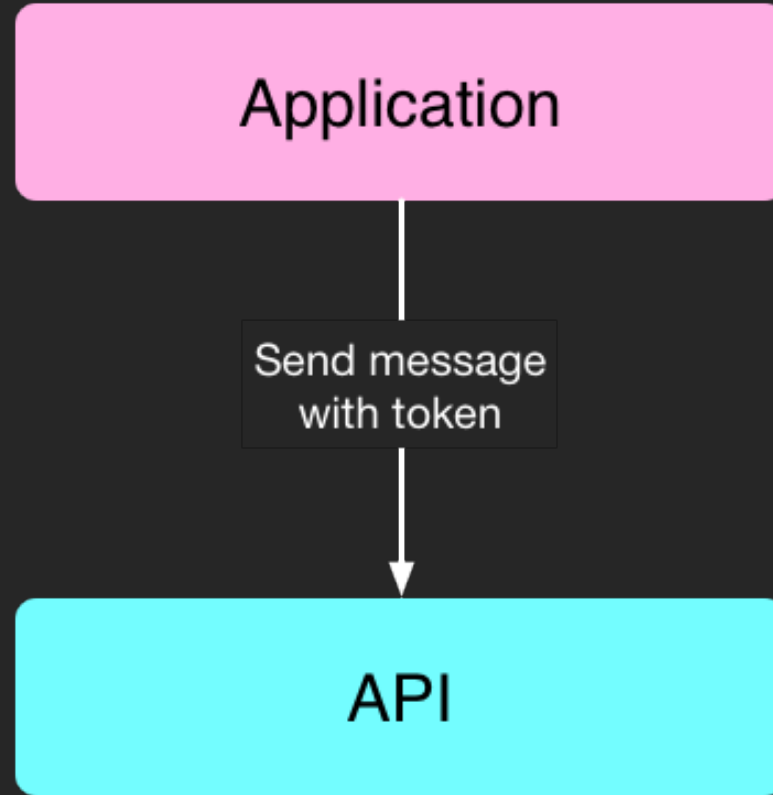
Authorization Code Flow



Authorization Code Flow



Authorization Code Flow

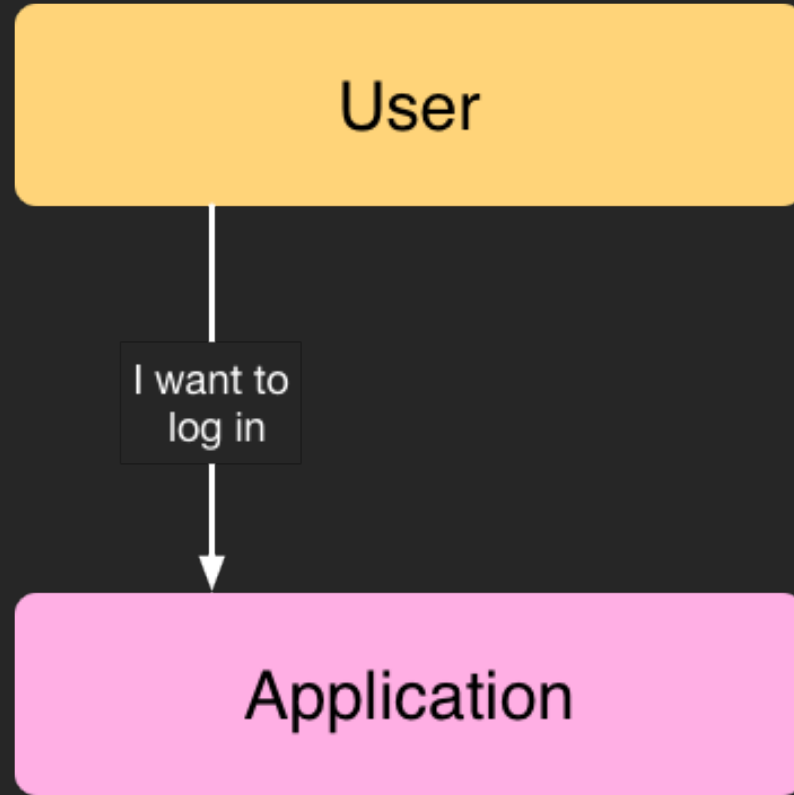


Implicit Flow

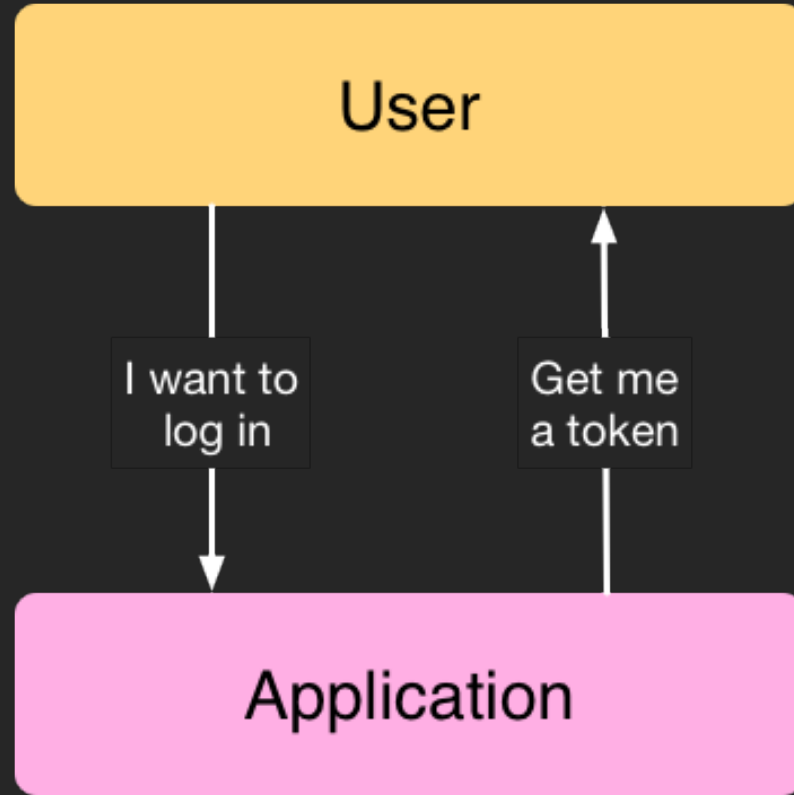
For logging into 3rd party apps and web SPAs



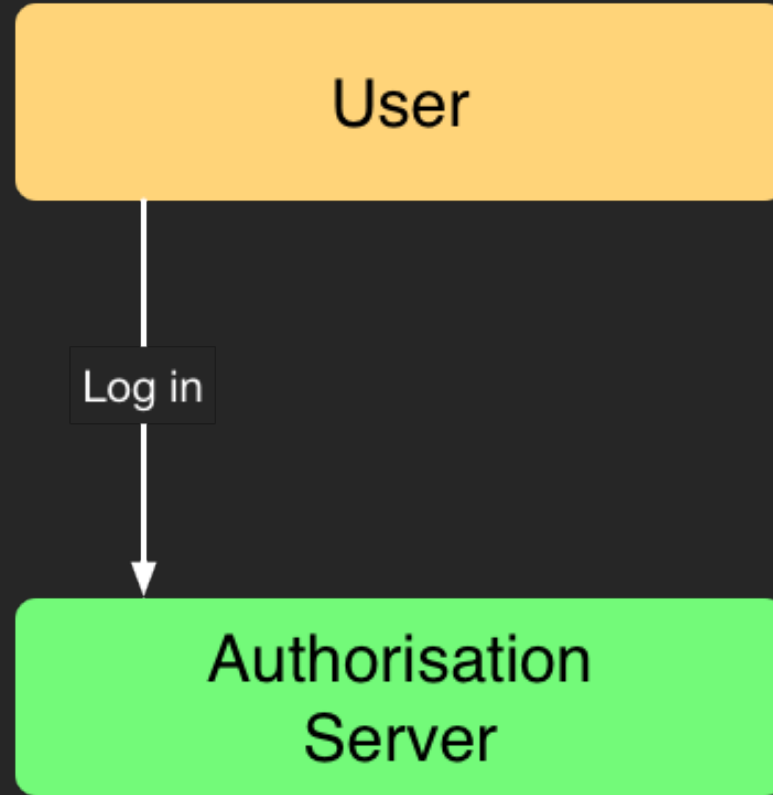
Implicit Flow



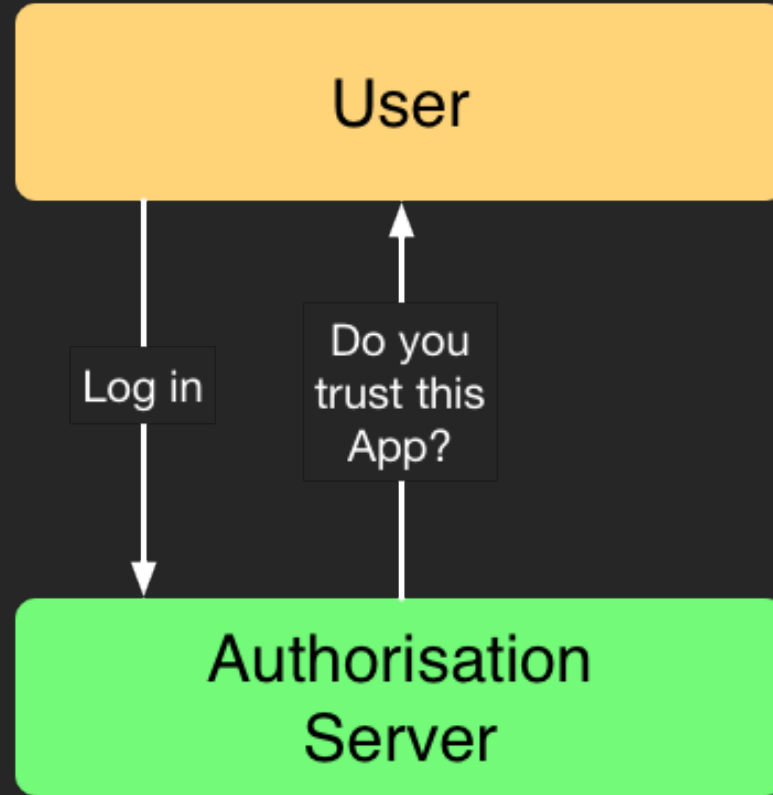
Implicit Flow



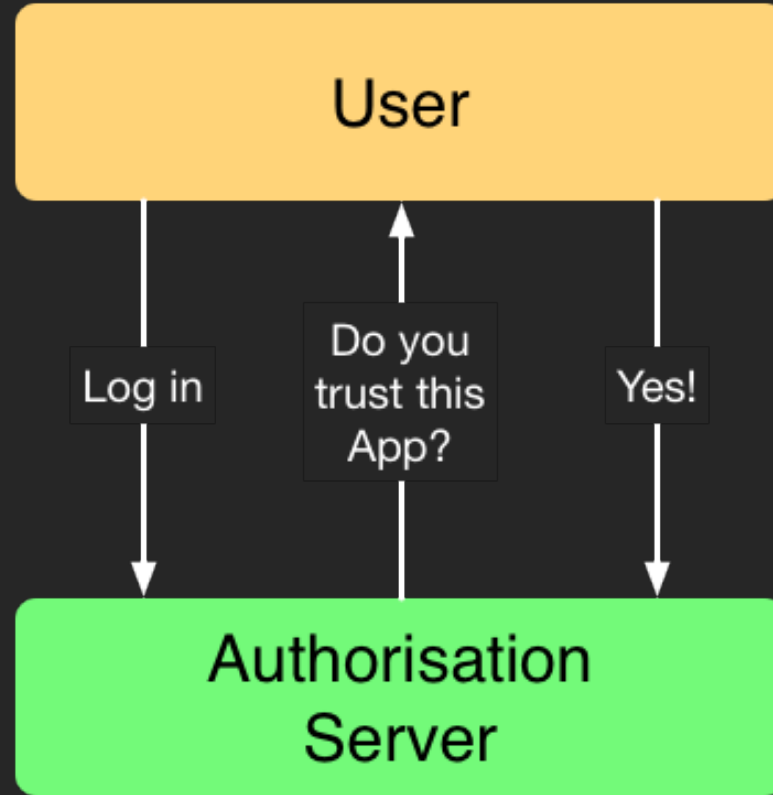
Implicit Flow



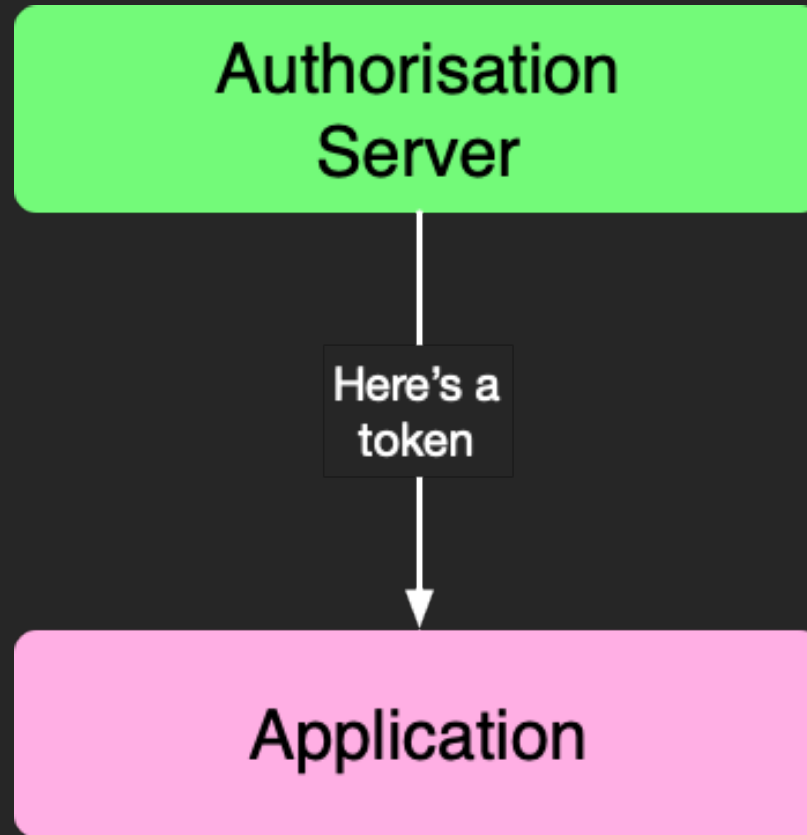
Implicit Flow



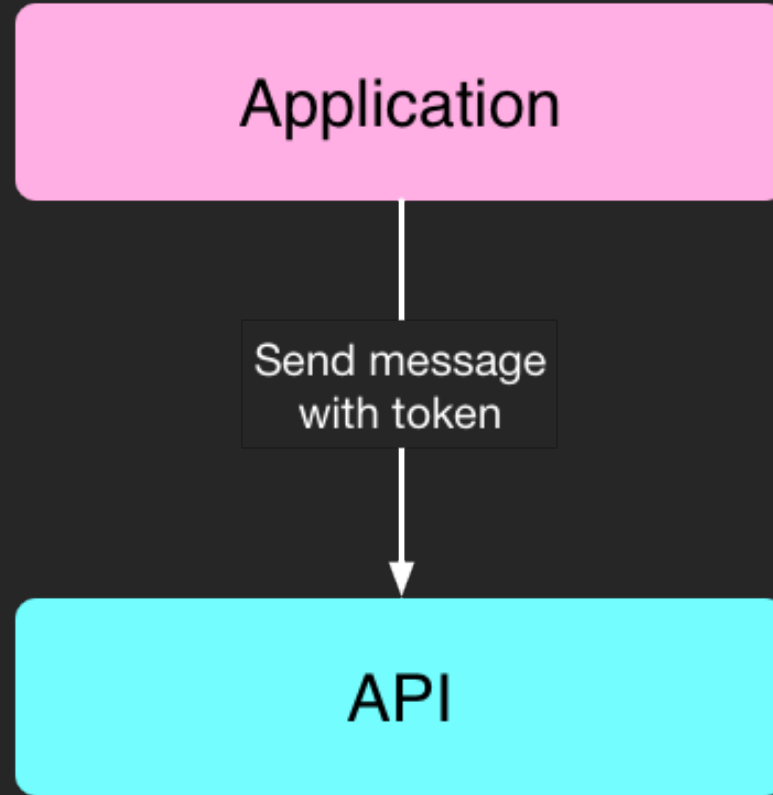
Implicit Flow



Implicit Flow



Implicit Flow

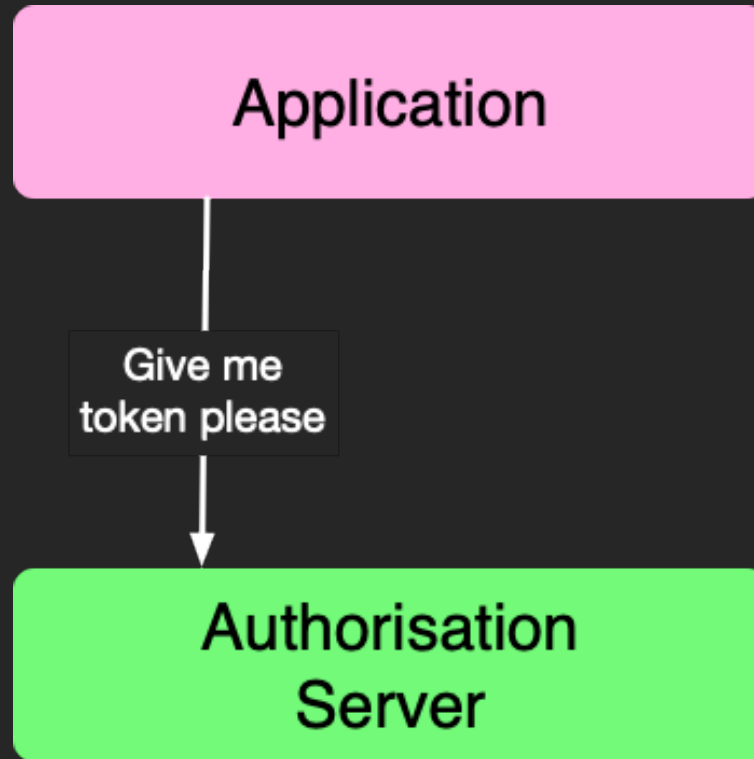


Client Credentials Flow

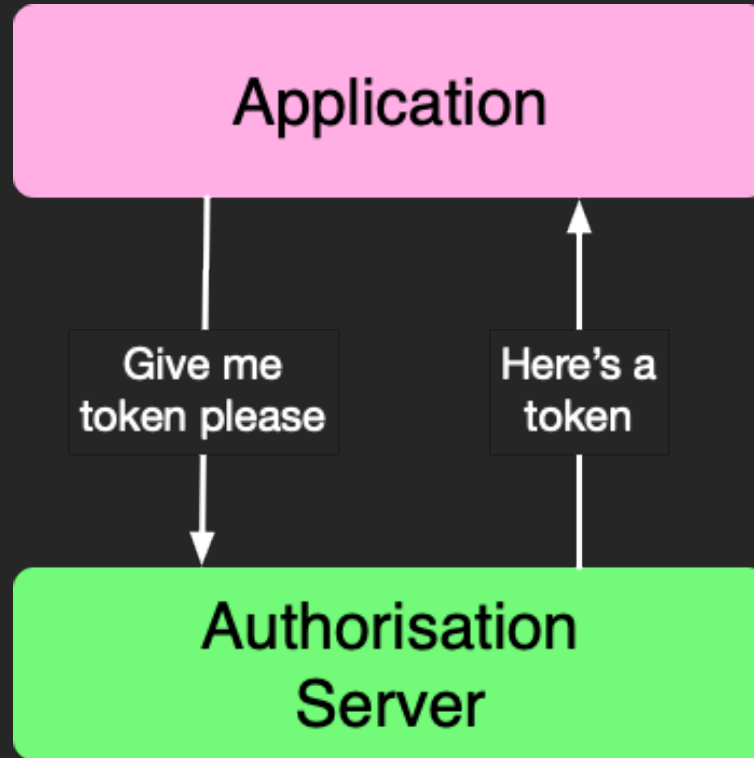
For jobs that don't need user permission



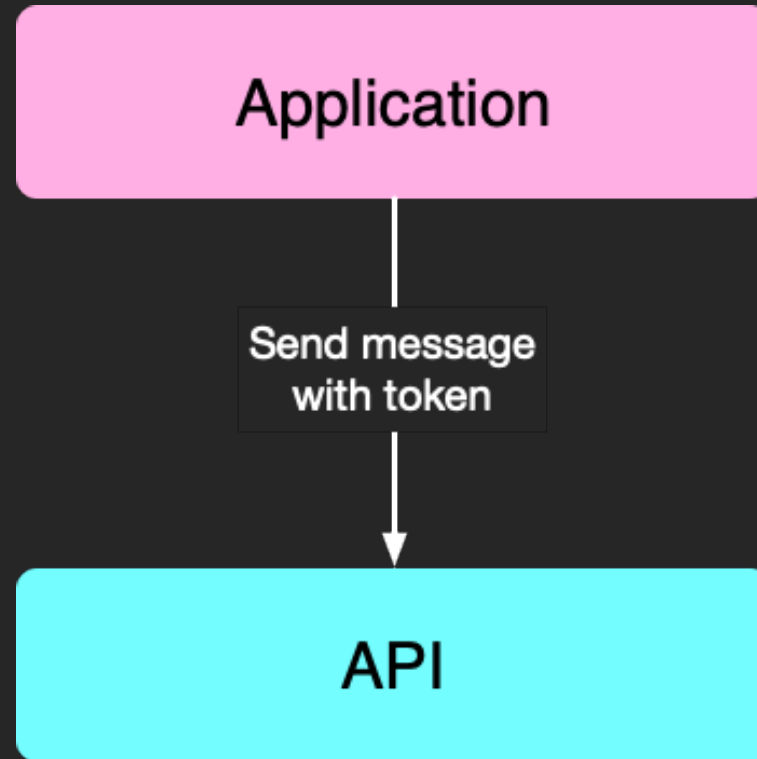
Client Credentials Flow



Client Credentials Flow



Client Credentials Flow



OAuth 2.0 Refresh token

- Allows the client to gain a new access token
- Refresh tokens need to be kept secure
- Authorization server can choose not to issue

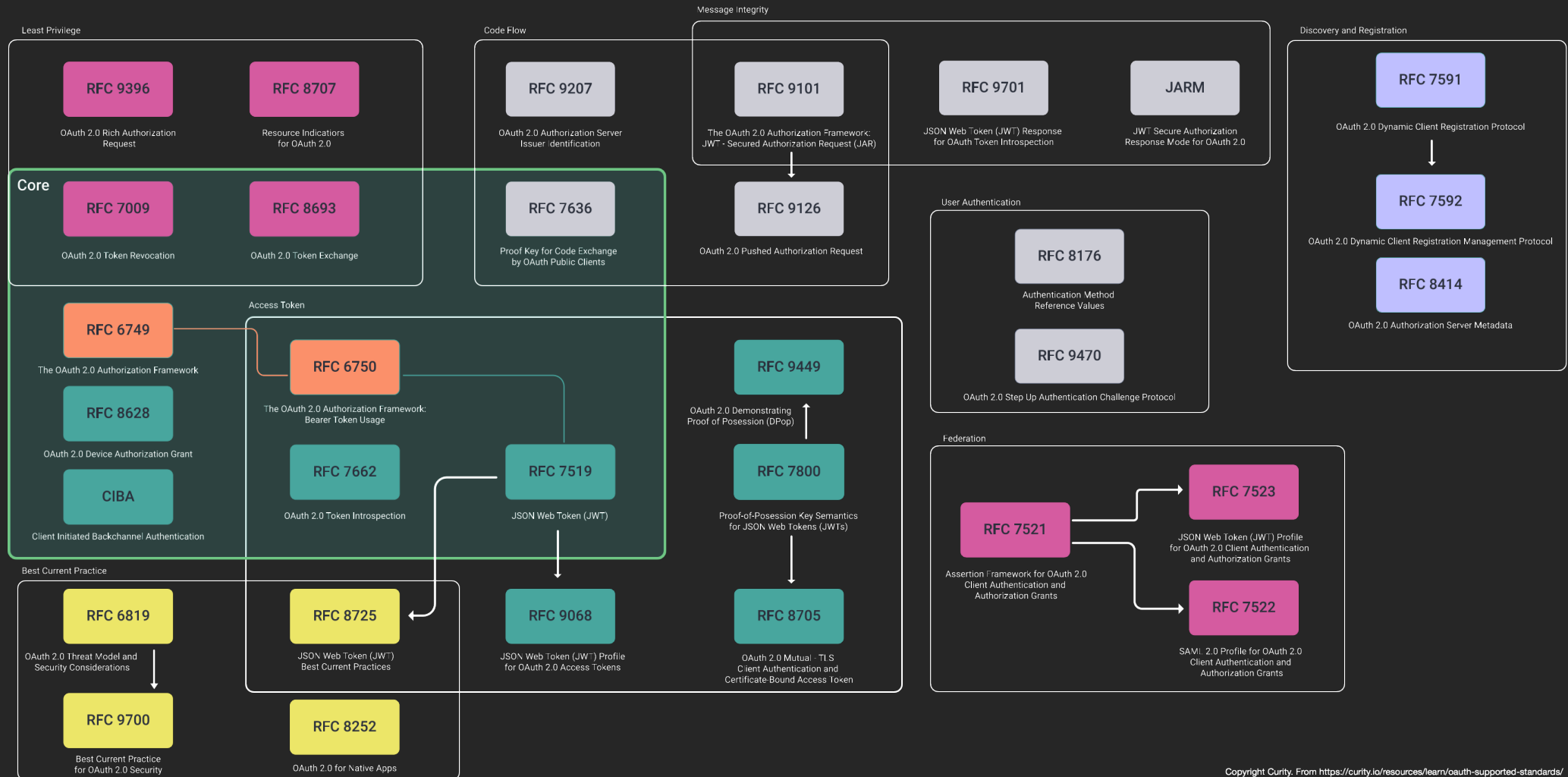


OAuth 2.0 Since 2012

The OAuth 2.0 Framework in 2012



The OAuth 2.0 Framework Today



Copyright Curity. From <https://curity.io/resources/learn/oauth-supported-standards/>

Key extensions since 2012

- RFC 7009: OAuth 2.0 Token Revocation
- RFC 7519: JSON Web Tokens (JWT)
- RFC 7636: Authorization Code without a client secret (PKCE)
- RFC 8628: Device Authorization grant for devices

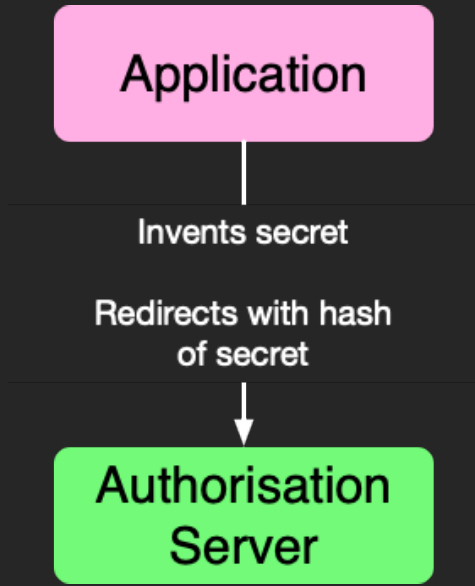


PKCE

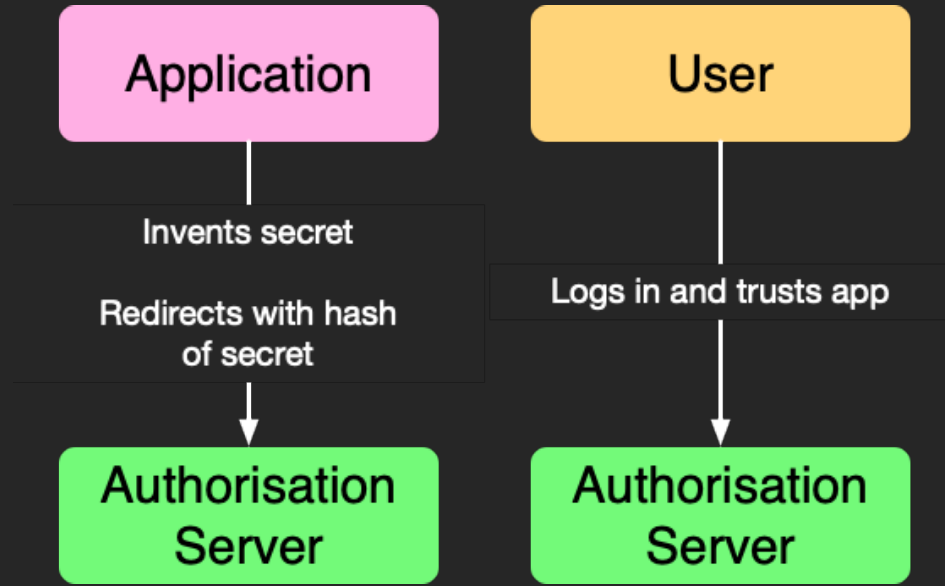
- First created for mobile, but useful for all public clients
- Protects the authorization code in the redirect
- We know that the right client is converting the code to a token



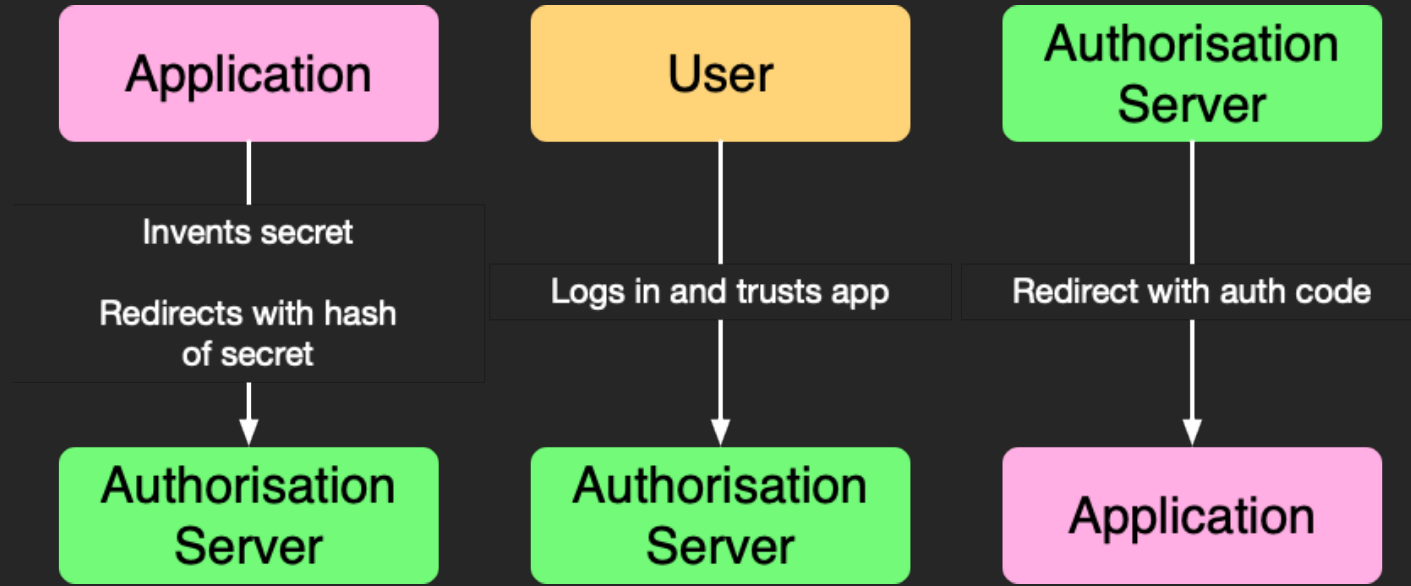
PKCE workflow



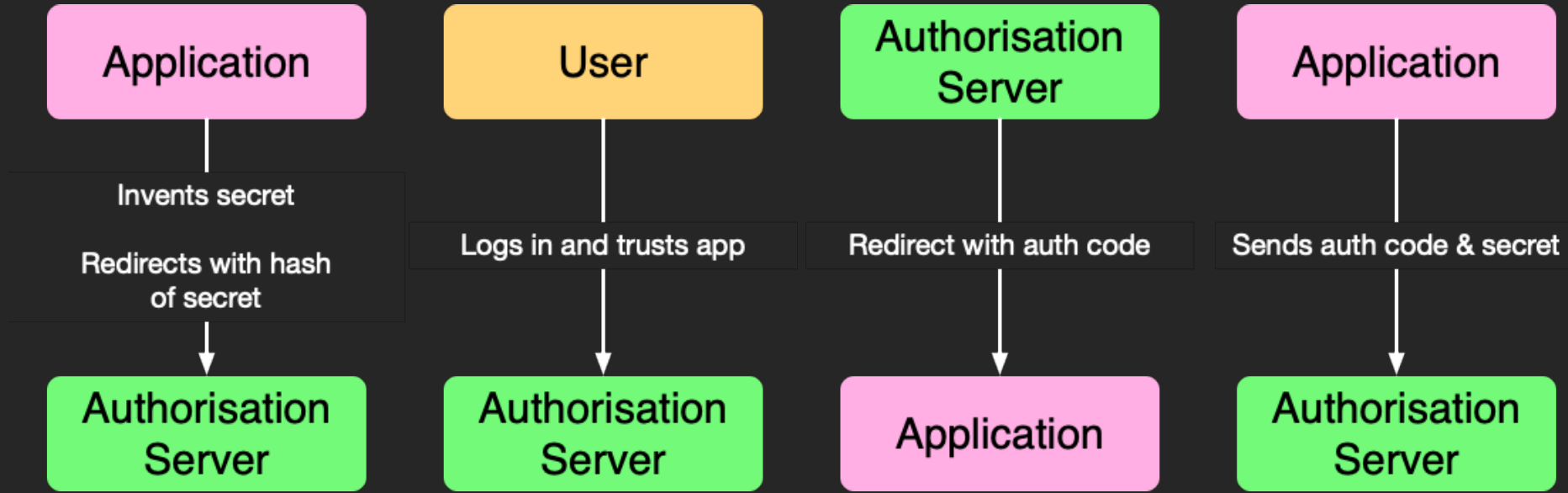
PKCE workflow



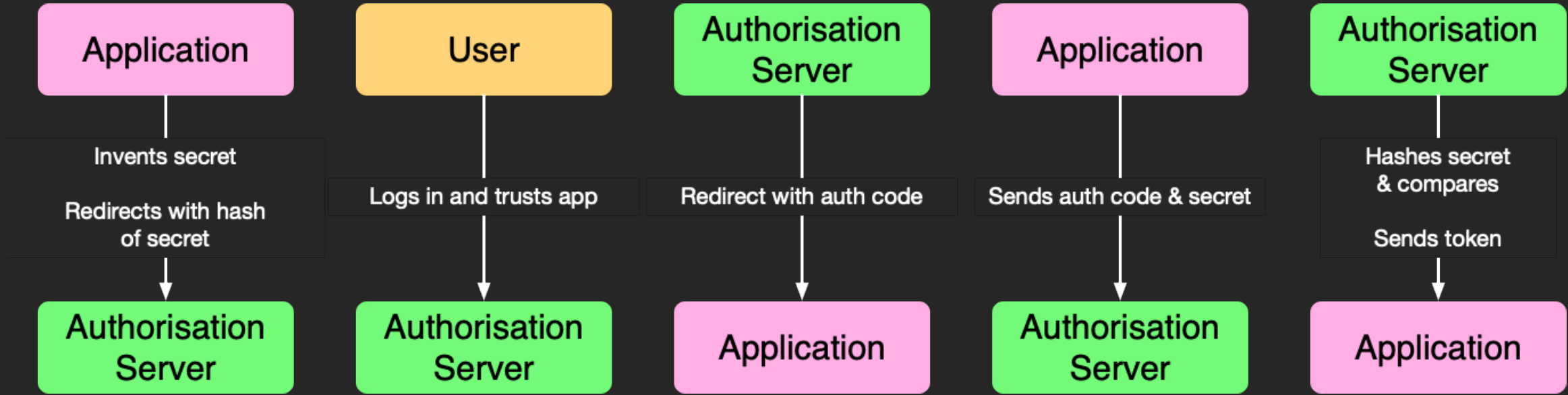
PKCE workflow



PKCE workflow



PKCE workflow

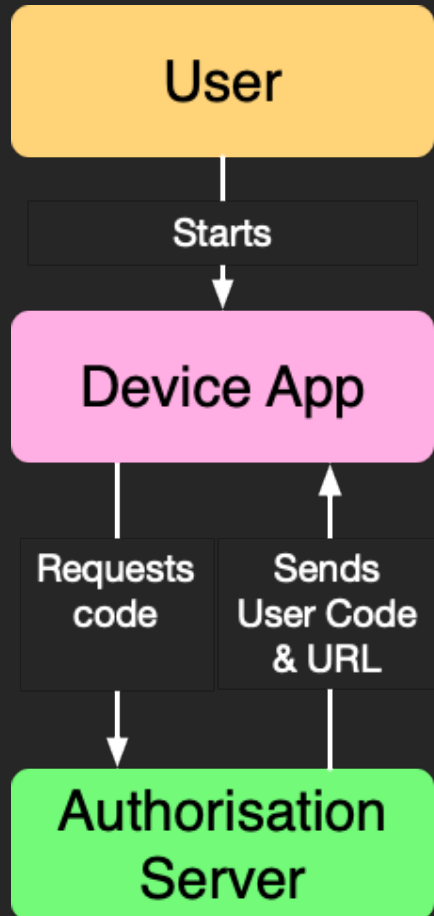


Device Authorization Flow

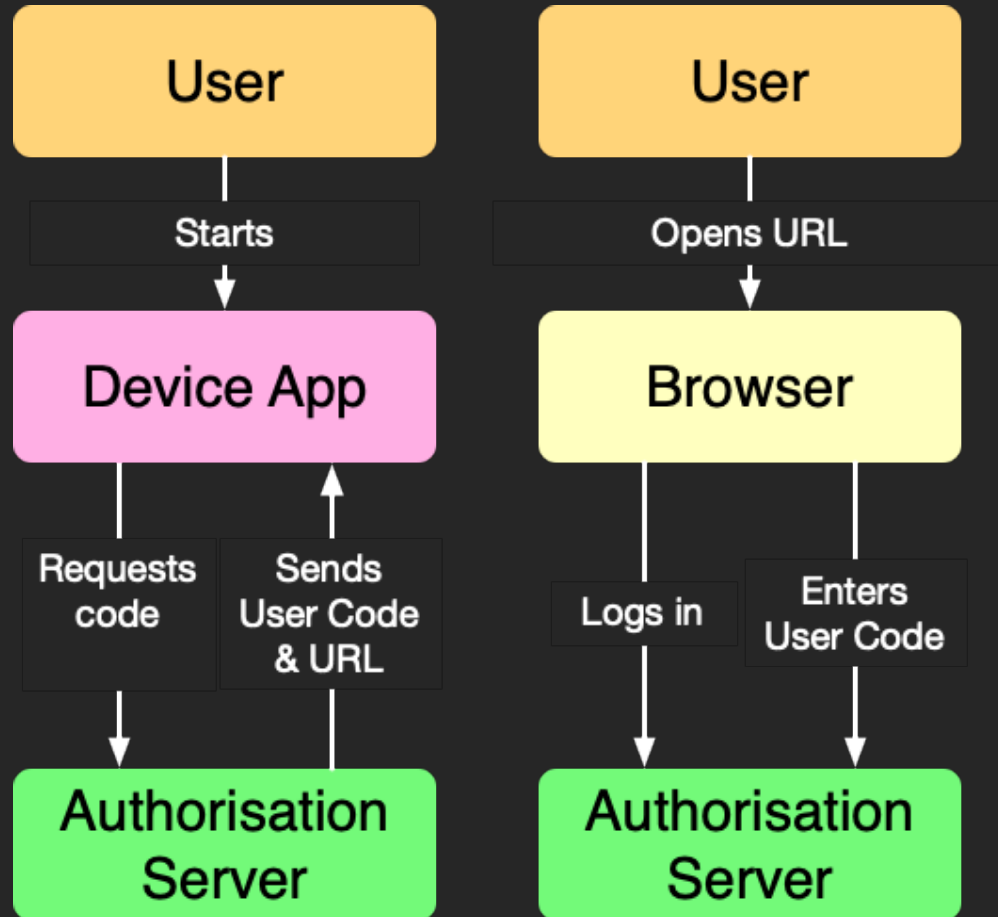
For apps with no browser (or keyboard)



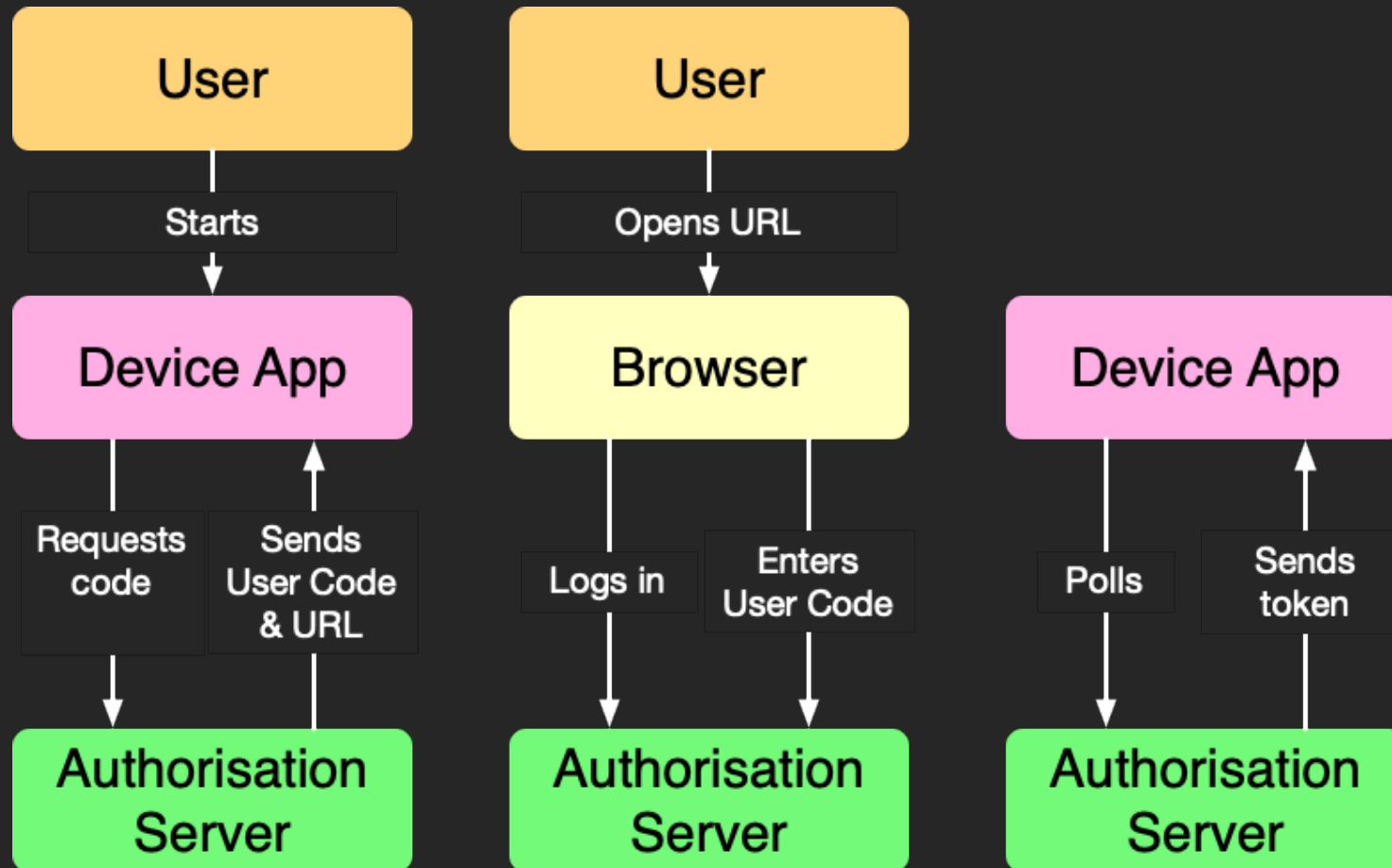
Device Authorization Flow



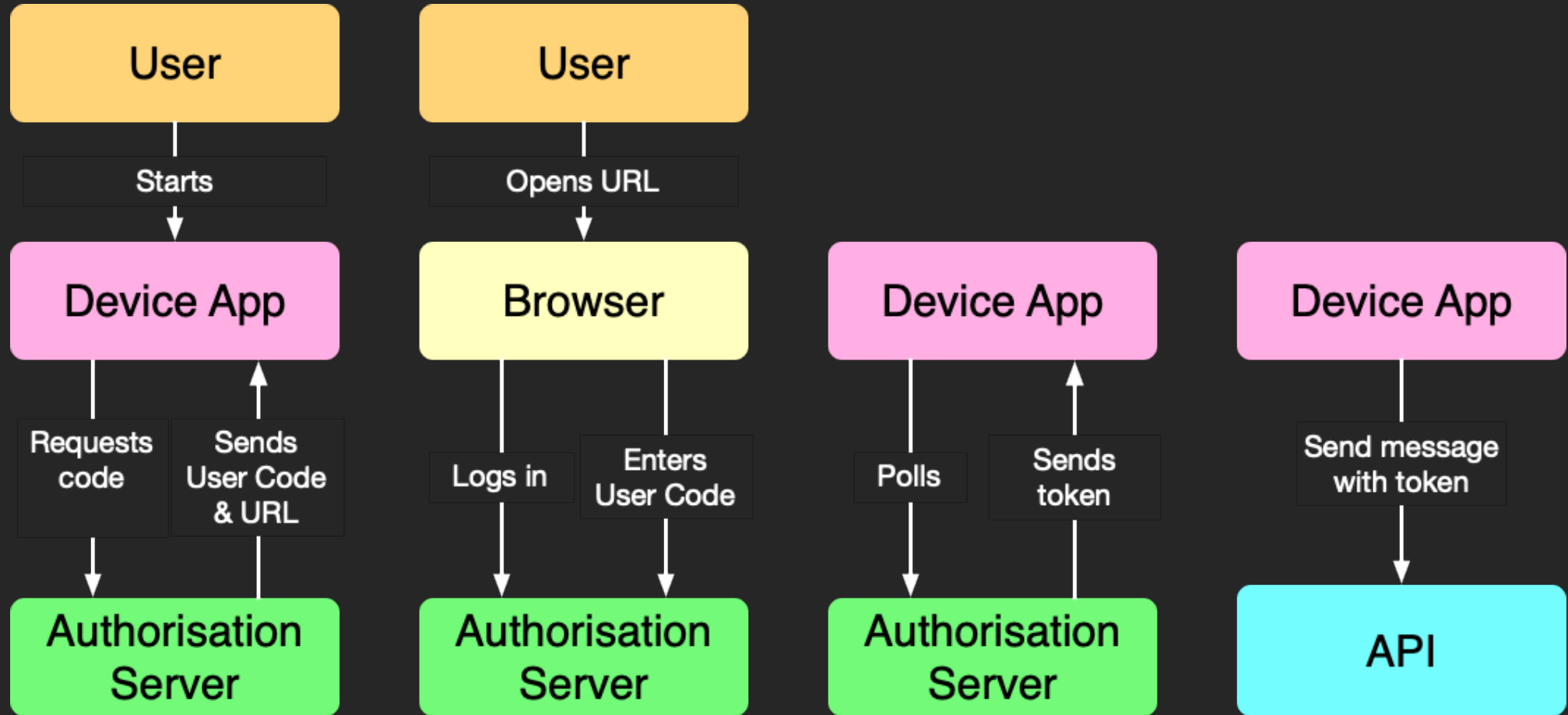
Device Authorization Flow



Device Authorization Flow



Device Authorization Flow



Best Practices since 2012

- RFC 7900: Best Current Practice for OAuth 2.0 Security
- RFC 8653: OAuth 2.0 for Native Apps
- OAUTH-WG: OAuth 2.0 for Browser Apps



OAuth 2.1

My main goal with OAuth 2.1 is to capture the current best practices in OAuth 2.0 as well as its well-established extensions under a single name.

Aaron Parecki

Key Objectives of OAuth 2.1

- Not a new protocol



Key Objectives of OAuth 2.1

- Not a new protocol
- Simplifies the specification



Key Objectives of OAuth 2.1

- Not a new protocol
- Simplifies the specification
- Incorporation of Best Current Practices



OAuth 2.1 Flows

- Authorization Code must be used with PKCE
- Device Authorization
- Client Credentials



Tokens

- No access tokens in query strings



Tokens

- No access tokens in query strings
- Refresh tokens must be sender constrained or one-time use



Tokens

- No access tokens in query strings
- Refresh tokens must be sender constrained or one-time use
- Refined token management (shorter lifetimes, rotation policies)



Other things

- Redirect URIs must be exact matches



Other things

- Redirect URIs must be exact matches
- State parameter is now mandatory for CSRF protection



Other things

- Redirect URIs must be exact matches
- State parameter is now mandatory for CSRF protection
- Confidential client now means a client that has credentials, otherwise it's Public



Going forwards

Migrating to 2.1

Migrating to 2.1

- Review current implementation

Migrating to 2.1

- Review current implementation
- Adopt Authorization Code flow with PKCE

Migrating to 2.1

- Review current implementation
- Adopt Authorization Code flow with PKCE
- Remove deprecated flows

Migrating to 2.1

- Review current implementation
- Adopt Authorization Code flow with PKCE
- Remove deprecated flows
- Secure redirect Uris

Migrating to 2.1

- Review current implementation
- Adopt Authorization Code flow with PKCE
- Remove deprecated flows
- Secure redirect Uris
- Review and implement the Best Current Practices

Other relatively new OAuth features



Other relatively new OAuth features

- Demonstrating Proof of Possession (RFC 9449)
- Mutual TLS (RFC 8705)



Other relatively new OAuth features

- Demonstrating Proof of Possession (RFC 9449)
- Mutual TLS (RFC 8705)
- JWT Profile for Access Tokens (RFC 9068)
- JWT-Secured Authorization Requests (RFC 9101)



Other relatively new OAuth features

- Demonstrating Proof of Possession (RFC 9449)
- Mutual TLS (RFC 8705)
- JWT Profile for Access Tokens (RFC 9068)
- JWT-Secured Authorization Requests (RFC 9101)
- Rich Authorization Requests (RFC 9396)
- Pushed Authorization Requests (RFC 9126)





Donatello @ HÖR BERLIN
@firebel.ly

+ Follow

OAuth 2.0? That's last years tech, I'm ready for 2.1!!!

8 May 2024 at 02:15

1 like



A rustic wooden fence made of horizontal logs is the central focus. A large, weathered metal padlock with a keyhole is attached to the fence with a chain. The padlock has a circular logo with the letters 'ACE' inside. The background is a soft-focus forest with trees showing autumn foliage in shades of yellow, orange, and green. The ground is covered in fallen brown leaves.

Thank you!