Standard Oauth2.0

2 browse init by User

User

3 redirect to AS with
- ClientID
- Callback path

4 redirect to AS by Client with
- client ID
- state
- scope
- response type
- callback uri

5 user credential

7 redirct to Client by AS with
- auth code
- state

0 client register :
- Client Name
redirect callback

1 client register :
- Client ID
- Client Secret

Client: App Server

Auth Server

6 redir to Client using callback path
- auth-code

8 verify "state"

9 exchange Access Token with
- authorization code
- client id
- client secret

10 access token

14 query AS introspection

11 request RS with access token

13 verify token, 2 ways:
(1) call AS introspection endpoint, or
(2) if JWT self-contained, using public KEY

Security Proxy
for Resource
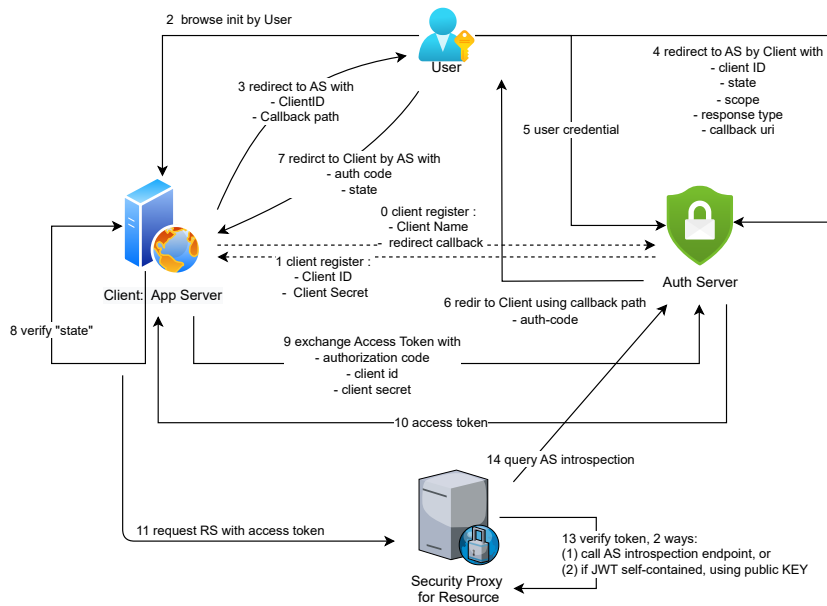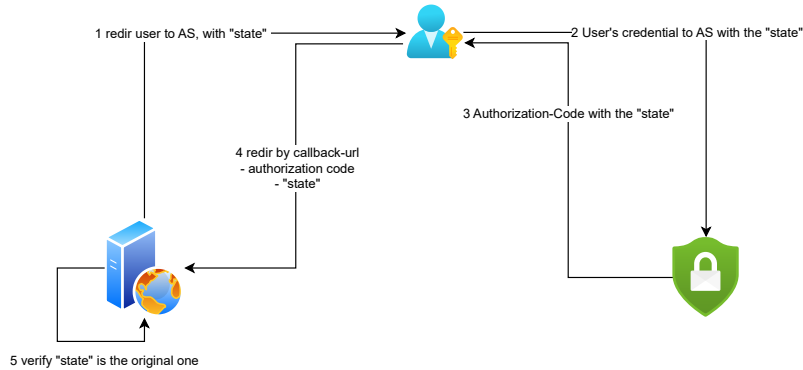
Authorization Code with State

When the AS redirects back to the client with the authorization code it will include the same
state parameter, and the client should check that it matches the original one sent on the
request. This ensures that the code received by the client is the one it requested. Otherwise, an
attacker may be able to craft a link that calls the client's redirect endpoint directly with an
authorization code obtained by the attacker.

1 redir user to AS, with "state"

2 User's credential to AS with the "state"

3 Authorization-Code with the "state"

4 redir by callback-url
- authorization code
- "state"

5 verify "state" is the original one

Refresh Token

A client can use a *refresh token* to obtain a fresh access token when the original one expires.
This allows an AS to issue short-lived access tokens without clients having to ask the user
for a new token every time it expires.

1 access token
refresh token

2 access token expired,
query with refresh token, ClientID, Client Secret

4 new acces token, new refresh token

Client

AS

3 verify the refresh token
is valid and used only once