# EXTENSION RFCS

- OAuth 2.0 Threat Model and Security Considerations (RFC6819)
- JSON Web Token (JWT) (RFC 7519)
- Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants (RFC 7521)
- Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants (RFC 7522)
- JSON Web Token (JWT) Profile For OAuth 2.0 Client Authentication and Authorization Grants (RFC 7523)
- OAuth 2.0 Dynamic Client Registration Protocol (RFC 7591)
- OAuth 2.0 Dynamic Client Registration Management Protocol (RFC 7592 EXP)
- Proof Key for Code Exchange by OAuth Public Clients (RFC 7636)
- OAuth 2.0 Token Introspection (RFC 7662)

# RFC7521 -ASSERTION FRAMEWORK FOR OAUTH 2.0 CLIENT AUTHENTICATION AND AUTHORIZATION GRANTS

## ➤ Using Assertions as Authorization Grants

- ➤ grant_type
- ➤ assertion
- ➤ scope

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Asaml2-bearer&
assertion=PHNhbWxwOl...[omitted for brevity]…ZT4
```

## ➤ Using Assertions for Client Authentication

- ➤ client_assertion_type
- ➤ client_assertion
- ➤ client_id

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=n0esc3NRze7LTCu7iYzS6a5acc3f0ogp4&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Asaml2-bearer&
client_assertion=PHNhbW...[omitted for brevity]...ZT
```

## RFC 7523- JSON WEB TOKEN (JWT) PROFILE FOR OAUTH 2.0 CLIENT AUTHENTICATION AND AUTHORIZATION GRANTS

- This specification defines the use of a JSON Web Token (JWT) Bearer Token as a means for requesting an OAuth 2.0 access token as well as for client authentication.

# PKCE (RFC 7636)

- The client creates and records a secret named the ”code_verifier" and derives a transformed version "t(code_verifier)" (referred to as the "code_challenge"), which is sent in the OAuth 2.0 Authorization Request along with the transformation method ”t_m".
- The Authorization Endpoint responds as usual but records "t(code_verifier)" and the transformation method.
- The client then sends the authorization code in the Access Token Request as usual but includes the "code_verifier" secret generated at (A).
- The authorization server transforms "code_verifier" and compares it to "t(code_verifier)" from (B). Access is denied if they are not equal.