Introduction

Purpose and scope

This runbook explains how a confidential client, such as a backend component or secure web application, can integrate with the Verifier acting as an Authorization Server (AS) using the Authorization Code Flow with `client_secret_jwt` client authentication. It provides a complete view of the flow — from the signed Authorization Request to token acquisition and usage — following OAuth 2.1 best practices.

Main aspects covered include:

- Integration of confidential clients with the Verifier using Authorization Code Flow + `client_secret_jwt`.
- Use of a signed JWT as the client authentication method instead of a static secret.
- OAuth 2.1 authorization_code profile with `request_uri` pointing to a signed Authorization Request Object.
- Token acquisition and usage for accessing Verifier-protected APIs.
- Security, signature validation, error handling, and observability.

Intended audience

- Developers integrating backend or confidential clients with the Verifier.
- Technical integrators configuring secure OAuth 2.1 clients.
- SRE and security engineers auditing token-based authentication.

High-level architecture

embedded-image-AuthCode-clientjwt-arch.png

1. The confidential client builds and signs an Authorization Request Object (JWT) and hosts it at a `request_uri`.
2. The client redirects the user to the Verifier Authorization Endpoint, including the `request_uri`.
3. The Verifier retrieves and validates the signed request object using the client's public key (`jwks_uri`).
4. The Verifier authenticates the user and returns an authorization code.
5. The client exchanges the code for tokens using `client_secret_jwt` authentication.
6. The Verifier issues access, ID, and refresh tokens.

High-level flow

embedded-image-AuthCode-clientjwt-flow.png

1. The client creates and signs a JWT containing the Authorization Request parameters, making it available at the provided `request_uri`.
2. The user is redirected to the Authorization Server, which retrieves and validates the JWT.
3. After successful authentication and consent, the AS issues an authorization code.
4. The client exchanges the code for tokens, authenticating with `client_secret_jwt`.
5. The AS validates the JWT and issues access, ID, and refresh tokens.
6. The client uses the access token to call protected APIs, and refreshes tokens as needed.

Revision #1
Created 10 November 2025 15:32:24 by Roger Miret
Updated 10 November 2025 15:42:21 by Roger Miret