**Flow Description**

1. **User "A" (Service/Script) calls application’s Auth API** with their credentials
2. **Application Auth API** uses its  client ID and secret to authenticate with the Identity Provider (IdP, say FED-SSO or PingFederate) using the **Client Credentials Grant**.
3. **Identity Provider** returns an access token to your Auth API.
4. **Your Auth API** verifies the token, then mints its own JWT, embedding User "A" as the subject.
5. **Your Auth API** returns the new JWT to User "A" as the API response.

**Flowchart**

**A diagram of a software application

Description automatically generated**

**Step-by-Step Flow**

1. **User "A"** sends a request to /api/auth/token on your Auth API.
2. **Auth API** uses its client credentials to request a token from the IdP.
3. **IdP** returns an access token.
4. **Auth API** verifies the access token.
5. **Auth API** creates a new JWT, setting the subject to User "A".
6. **Auth API** returns this JWT to User "A".

Once your Auth API mints a custom **JWT** (e.g., with "User A" as the subject), **any downstream API** that receives this token must be able to:

**Validate the Minted JWT Before Allowing Access**

**Validation Steps in the Consumer API (Downstream API):**

1. **Parse the JWT**
2. **Verify the signature** using the shared secret (or public key if using RSA)
3. **Check standard claims**:
   * exp – expiration time
   * iss – issuer
   * sub – subject (e.g., "User A")
4. **Authorize based on claims** (e.g., roles or permissions)

**HMAC (HS256) – Shared Secret**

* Consumer API must use the **same secret key** used by Auth API (supersecretkey1234567890)
* Use a JWT library like Nimbus, Auth0, or Spring Security

**Java Code (Nimbus example in consumer API):**

SignedJWT jwt = SignedJWT.parse(token);

JWSVerifier verifier = new MACVerifier("supersecretkey1234567890");

if (!jwt.verify(verifier)) {

throw new SecurityException("Invalid signature");

}

// Check expiration

Date exp = jwt.getJWTClaimsSet().getExpirationTime();

if (exp.before(new Date())) {

throw new SecurityException("Token expired");

}

// Check subject

String user = jwt.getJWTClaimsSet().getSubject();

// Continue with user-based authorization