

Eugene Auth — End-to-End Flow

Developer walkthrough: MS Entra ID Authorization Code flow → MSAL token exchange → Eugene HS256 JWT mint → HTTPBearer dependency → per-request claim validation

Audience

Engineers who need to understand how a Eugene API caller obtains an access token and how every protected route verifies that token. The auth router exposes four endpoints: an Entra ID login redirect, an OIDC callback that exchanges the authorization code for an Entra access token and mints a Eugene JWT, a JSON token endpoint for local SPA development, and a `/auth/whoami` debug probe. Every reference uses the form `path/to/file.py:line`.

Reference endpoints

- `GET /login` — either redirects to Microsoft Entra ID (production) or short-circuits to a local-development token (when `ENVIRONMENT == 'local'`).
- `GET {REDIRECT_PATH}` — the OIDC redirect URI; Microsoft posts the authorization code here and Eugene exchanges it for tokens.
- `POST /auth/token` — JSON access token for programmatic / Next.js SPA clients. Local mode only.
- `GET /auth/whoami` — returns the decoded Eugene JWT claims; the canonical example of `Depends(get_current_user)`.

Caveat

The Eugene token is **HS256**-signed with a shared secret (`EUGENE_CLIENT_SECRET`) — this is a first-party JWT used purely for authorization inside Eugene, not an Entra-issued token. The Entra access token never leaves the auth callback; it is decoded once to lift the user's `upn / tid / sub` claims and is then discarded in favor of the freshly minted Eugene token. See **RS256 vs HS256** — rotating the shared secret invalidates every live Eugene token.

0. Auth model (read this first)

Two token types are in play:

Entra ID access token (RS256, issued by Microsoft)

```
issuer:  https://login.microsoftonline.com/<tenant_id>/v2.0
audience:  ENTRA_CLIENT_ID
algorithm: RS256 (verified via JWKS for ID tokens)
lifetime:  controlled by Entra (~60-90 min)
scope:     ENTRA_SCOPE (env-driven, comma-separated)
```

Eugene access token (HS256, issued by this service)

```
issuer:  https://eugene.ai.cslg1.cslg.net/<EUGENE_TENANT_ID>
audience:  api://eugene/<EUGENE_CLIENT_ID>
algorithm: HS256 (shared secret: EUGENE_CLIENT_SECRET)
lifetime:  EUGENE_TOKEN_TTL_SECS = 24 * 60 * 60 (24h)
claims:    iss, aud, iat, exp, tid, sub, upn, oid, name,
           preferred_username, roles
```

- All values are pinned in `src/router/auth/const.py:44-48`.
- Two roles are defined in `const.py:51-52`: `user.public.read` and `user.confidential.read`.
- Role mapping is a static dict keyed by `tenant_id|upn` in `src/router/auth/roles.py:17-31`. Unknown users fall through to `DEFAULT_ROLES = {user.public.read}`.
- The Eugene token carries the Entra `tid / sub / upn / oid` claims verbatim so downstream services can trace requests back to the Entra principal.

1. HTTP entry — the auth router

`src/router/auth/auth_router.py`

- Router is registered in `src/eugene_ws.py:32,56` via `from router.auth import auth_router` and an `app.include_router(auth_router, ...)`.
- Prefix is the empty string (`prefix=''`, line 24) so paths are mounted at the root.
- Tag `'auth'` groups the endpoints under one Swagger section (line 25).

GET /login (line 29)

Branches on `ENVIRONMENT`: when it is `'local'` or `MASL_APP` is `None`, the handler mints a local development Eugene token and returns it embedded in an HTML page (so the developer can copy/paste it into Swagger). In production it asks MSAL for an authorization URL and 302s the browser to Microsoft.

```
auth_url = MASL_APP.get_authorization_request_url(
    scopes=ENTRA_SCOPE,
    redirect_uri=REDIRECT_URI,
    response_mode='query',
)
return RedirectResponse(auth_url)
```

GET {REDIRECT_PATH} — auth_callback (line 83)

- Reads `request.query_params`; returns 401 if Microsoft passed back an error or if code is missing.
- Calls `MASL_APP.acquire_token_by_authorization_code(code, scopes, redirect_uri)` to swap the code for an Entra payload.
- Hands the payload to `_validate_or_raise()` (line 120), which calls `decode_entra_id_token` for full JWKS-backed validation.
- Calls `entra_token_to_eugene_token(entra_token)` — this is where the Eugene JWT is minted.
- Returns an HTML page containing both tokens (lines 136-147); designed for developer copy/paste, not browser session use.

POST /auth/token (line 59)

JSON endpoint for SPA / programmatic clients. Refuses with HTTP 403 if `ENVIRONMENT != 'local'` — production must walk the redirect flow. The shape is the conventional OAuth-style envelope:

```
{
  "access_token": "<eugene-jwt>",
  "token_type": "Bearer",
  "expires_in": 86400
}
```

GET /auth/whoami (line 54)

The only protected endpoint in this router. Declares `user: dict = Depends(get_current_user)` — FastAPI resolves the dependency, which runs JWT validation, and the handler simply echoes the claims as JSON.

2. Token validation internals

[src/router/auth/auth.py](#)

`get_current_user` is a 5-line FastAPI dependency: it pulls `HTTPAuthorizationCredentials` from the `HTTPBearer` security scheme (declared as `SECURITY` in `conf.py:10`) and forwards the raw token to `validate_eugene_access_token`:

```
def get_current_user(
    credentials: HTTPAuthorizationCredentials = Depends(SECURITY),
):
    token = credentials.credentials
    return validate_eugene_access_token(token)
```

[src/router/auth/eugene_jwts.py](#)

`validate_eugene_access_token` (line 22) is the workhorse. It uses `python-jose` (from `jose import jwt`) to decode and verify the HS256 signature, then enforces a small set of required claims:

```
claims = jwt.decode(
    token=token,
    key=EUGENE_CLIENT_SECRET,
    algorithms=[EUGENE_TOKEN_ALGORITHM],    # 'HS256'
    audience=EUGENE_TOKEN_AUDIENCE,
    issuer=EUGENE_TOKEN_ISSUER,
    options={
        'require_exp': True,
        'require_iat': True,
        'verify_aud': True,
        'verify_iss': True,
    },
)

required_claims = ['tid', 'sub', 'roles', 'upn']
missing = [c for c in required_claims if c not in claims]
if missing:
    raise HTTPException(status_code=401,
                        detail='Invalid token structure')
```

- `ExpiredSignatureError` → 401 'Token expired'.
- `JWTErrors` (bad signature, wrong audience, wrong issuer, malformed) → 401 'Invalid token'.
- Any other exception → 500 'Authentication error' (logged as `logger.error`).
- On success the full claim dict is returned to the handler; FastAPI injects it as `user: dict`.

Entra ID token validation

[src/router/auth/entra_id_jwts.py](#)

Used during the OIDC callback only. `decode_entra_id_token` (line 63) uses `PyJWT` (`import jwt` — note the name collision with `python-jose`) to verify the ID token with RS256 against the Microsoft JWKS. `decode_entra_access_token` (line 87) deliberately skips signature verification (`options={'verify_signature': False}`) — the comment notes Entra access tokens cannot be verified locally, only their claims are lifted.

3. Response model — token / user payload

Eugene token claims (the dict returned by `validate_eugene_access_token`)

```
{
  "iss": "https://eugene.ai.cslg1.cslg.net/<EUGENE_TENANT_ID>",
  "aud": "api://eugene/<EUGENE_CLIENT_ID>",
  "iat": 1715900000,
  "exp": 1715986400,
  "tid": "<entra-tenant-id>",
  "sub": "<entra-subject-id>",
  "upn": "Damian.Knopp@cslbehiring.com",
  "oid": "<entra-object-id>",
  "name": "Damian Knopp",
  "preferred_username": "Damian.Knopp@cslbehiring.com",
  "roles": ["user.public.read", "user.confidential.read"]
}
```

/auth/whoami response

```
{
  "user": {
    "iss": "...",
    "aud": "...",
    "tid": "...",
    "sub": "...",
    "upn": "Damian.Knopp@cslbehiring.com",
    "roles": ["user.public.read"]
  }
}
```

/auth/token response

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIs...<base64>...",
  "token_type": "Bearer",
  "expires_in": 86400
}
```

Token minting (for completeness)

`_issue_internal_eugene_token` (`eugene_jwt.py:124`) is the only place Eugene tokens are signed:

```
payload = {
    'iss': EUGENE_TOKEN_ISSUER,
    'aud': EUGENE_TOKEN_AUDIENCE,
    'iat': now,
    'exp': now + timedelta(seconds=EUGENE_TOKEN_TTL_SECS),
    **claims, # tid, sub, upn, oid, name, preferred_username, roles
}
return jwt.encode(
    claims=payload,
    key=EUGENE_CLIENT_SECRET,
    algorithm=EUGENE_TOKEN_ALGORITHM, # 'HS256'
)
```

4. Dependencies — env, MSAL, JWKS cache

`src/router/auth/conf.py`

Every parameter is loaded from `os.environ` via `_env_value()` (line 84) — missing values raise `ValueError` at import time, so the service fails fast on misconfiguration.

Required environment variables

```
ENVIRONMENT           # 'local' enables the dev shortcuts
ENTRA_CLIENT_ID        # Entra application (client) id
ENTRA_CLIENT_SECRET    # Entra application client secret
ENTRA_TENANT_ID        # Entra directory (tenant) id
ENTRA_AUTHORITY        # https://login.microsoftonline.com/<tenant>
ENTRA_SCOPE            # comma-separated, split() in conf.py:55-57
REDIRECT_PATH          # e.g. /auth/callback
REDIRECT_URI           # full callback URL Microsoft posts back to
EUGENE_CLIENT_ID       # used in audience: api://eugene/<id>
EUGENE_CLIENT_SECRET   # HS256 signing key for Eugene tokens
EUGENE_TENANT_ID       # used in issuer + role map key
```

Derived constants

- `ENTRA_ISSUER =`
`f'https://login.microsoftonline.com/{ENTRA_TENANT_ID}/v2.0'` (const.py:28).
- `ENTRA_JWKS_URI` is fetched from `<issuer>/.well-known/openid-configuration` at import time (conf.py:66-68) — skipped in local mode.
- `MASL_APP` is a `ConfidentialClientApplication` from the `msal` Python library; `None` in local mode (const.py:34-38).
- `EUGENE_TOKEN_ISSUER =`
`f'https://eugene.ai.cslg1.cslg.net/{EUGENE_TENANT_ID}'` and
`EUGENE_TOKEN_AUDIENCE = f'api://eugene/{EUGENE_CLIENT_ID}'` (const.py:44, 48).

JWKS key fetching and caching

`entra_id_jwts.py` keeps an in-process dict (`_jwks_cache`, line 17) keyed by JWKS URI with an **8 hour** TTL (`_jwks_cache_ttl_secs = 60 * 60 * 8`, line 16). `fetch_entra_jwks()` (line 20) checks freshness, refreshes via `requests.get`, and falls back to the stale cache on network failure. `fetch_entra_signing_key()` (line 47) reads the unverified kid from the token header and matches it against the cached JWKS, materialising the public key via `RSAAAlgorithm.from_jwk(key)`.

Libraries in play

- `python-jose` (from `jose import jwt`) — used for the Eugene HS256 token (mint & verify) in `eugene_jwts.py`.
- `PyJWT` (`import jwt in entra_id_jwts.py`) — used for RS256 verification of Entra ID tokens and for unverified base64 decoding of Entra access tokens.
- `msal` — `ConfidentialClientApplication` for Authorization Code flow.
- `fastapi.security.HTTPBearer` — declared once as `SECURITY` in `conf.py:10` and reused by every protected route across the service.

5. Suggested debugging walk

- Set `ENVIRONMENT=local` in `.env` and bring the service up: `docker-compose up eugene_ws`. With `MASL_APP=None`, `/login` short-circuits to a local token — useful when Entra is not reachable from your laptop.
- Curl the JSON variant: `curl -X POST http://localhost:8000/auth/token`. Paste the `access_token` into `https://jwt.ms` to inspect `iss` / `aud` / `exp` / `roles`.
- Hit `GET /auth/whoami` with `Authorization: Bearer <token>` and breakpoint at `src/router/auth/eugene_jwt.py:28` to watch `jwt.decode` verify the signature, audience, and issuer in one call.
- To force the production path, unset `ENVIRONMENT` (or set it to anything other than `'local'`) and ensure the Entra env vars are populated. Hit `/login` in a browser; you should be 302'd to `login.microsoftonline.com`. Microsoft then redirects to `REDIRECT_URI` with `?code=...`
- Breakpoint at `auth_router.py:98` (the `acquire_token_by_authorization_code` call) to inspect Entra's response — you should see `access_token`, `id_token`, and `id_token_claims`. Step into `entra_token_to_eugene_token` (`eugene_jwt.py:63`) to watch the `upn/tid/sub` lift, the role lookup, and the HS256 mint.
- To exercise role mapping, add your `upn` to `USER_ROLE_MAP` in `src/router/auth/roles.py:17-31` and confirm the `roles` claim on the next token includes `user.confidential.read`. Unknown users get `DEFAULT_ROLES` only.

6. Reference index

HTTP entry

```
src/router/auth/auth_router.py
/login                line 29
/auth/whoami          line 54
/auth/token           line 59
{REDIRECT_PATH}       line 83 (auth_callback)
_validate_or_raise    line 120
_generate_token_html  line 136
```

Dependency (FastAPI Depends)

```
src/router/auth/auth.py      get_current_user
src/router/auth/conf.py:10    SECURITY = HTTPBearer(...)
```

Token validation + mint

```
src/router/auth/eugene_jwt.py
validate_eugene_access_token      line 22
entra_token_to_eugene_token       line 63
issue_local_development_eugene_token_with_roles line 96
_issue_eugene_token_with_roles    line 113
_issue_internal_eugene_token       line 124
```

Entra ID JWKS / decode

```
src/router/auth/entra_id_jwt.py
fetch_entra_jwks      line 20 (8h TTL cache)
fetch_entra_signing_key line 47
decode_entra_id_token line 63 (RS256 + JWKS)
decode_entra_access_token line 87 (no signature verify)
```

Config + env

```
src/router/auth/conf.py
src/router/auth/const.py
```

Roles

```
src/router/auth/roles.py
USER_ROLE_MAP      line 17
DEFAULT_ROLES      line 33
load_roles_for_user line 36
load_roles_for_local_development line 63
```

Registration

```
src/eugene_ws.py:32,56 include_router(auth_router, ...)
```