ChatGPT

Appendix D – APIs & SDKs

Public API overview

The wallet platform exposes a set of RESTful APIs documented in the accompanying **OpenAPI 3.0** specification (`/deliverables/apis/public.openapi.yaml`). All API calls require authentication via OAuth2/OIDC; citizen-facing flows use the PAR + PKCE pattern, while administrative and issuer APIs require client credentials and mTLS. Below is a high-level summary of the key endpoints. Full details (request/response schemas, error codes, examples) are provided in the OpenAPI file.

Path & method	Purpose	Auth & standards
`POST /api/credentials/issue`	Issuance endpoint to create a new verifiable credential. Accepts an issuance request object (schema ID, subject DID, attributes) and returns a signed credential envelope and status URL.	Client authenticates via OID4VCI (`/par` and `/token` endpoints) and obtains an access token; issuer must have appropriate role.
`GET /api/credentials/{id}/status`	Returns the current status (valid, suspended, revoked) of a credential. Used by wallets to poll for updates or by verifiers to check revocation.	Requires bearer token scoped to the credential's tenant; status list URLs may be publicly cacheable.
`POST /api/presentations/verify`	Verifies a credential presentation (JWT/JSON-LD) and returns a verification result (validity, issuer trustworthiness, disclosed attributes).	Verifier SDK obtains an access token via OIDC4VP and submits the presentation; the API validates signatures, checks revocation lists and applies selective disclosure.
`POST /api/revocations`	Allows an issuer or authorised administrator to revoke or suspend a credential. A revocation reason and effective date are captured, and status lists are updated.	Requires client credentials and mTLS; only authorised roles may revoke.
`GET /api/trust/issuers`	Returns the list of trusted issuers and their certificate chains. Enables verifiers and third parties to maintain local caches of trust lists.	Public endpoint with optional API key; data signed by PKI service.
`POST /api/tenants`	Administrative endpoint to onboard a new tenant. Accepts configuration (name, region, policies, PKI parameters) and provisions a new environment.	Requires administrator role and mTLS.

Path & method	Purpose	Auth & standards
`GET /api/metrics`	Returns consumption metrics such as number of credentials issued, presented and revoked, active wallets, API latency, etc. Supports time-series queries for consumption reporting.	Requires operator role; aggregated metrics may also be exposed to DGov dashboards.

OID4VCI / OIDC4VP flows

The issuance and presentation APIs follow the **OpenID for Verifiable Credential Issuance (OID4VCI)** and **OpenID for Verifiable Presentations (OIDC4VP)** standards. The `/par` endpoint allows the ServiceWA app to submit a pre-authorisation request; the user authenticates and consents via an authorisation page hosted by the wallet provider, and the app exchanges the authorisation code for an access token. The issued credential is returned out-of-band in a secure channel (e.g., wallet binding API). For presentations, the verifier creates a presentation request specifying disclosure requirements; the citizen's app constructs a verifiable presentation and sends it along with the access token to `/api/presentations/verify`.

SDKs

We provide SDKs to simplify integration for different platforms. All SDKs share common design principles: they encapsulate the OIDC/OAuth flows, handle cryptographic operations (key generation, signing, verification) via the device's secure storage, manage local caching and offline presentation, and expose simple methods for issuance and presentation.

Flutter SDK (`/deliverables/sdk/flutter`)

- **Purpose:** Embeds wallet functionality into the **ServiceWA** app. Supports both Android and iOS via a single codebase.
- **Features:**
- Handles OIDC authentication via PKCE and dynamic client registration.
- Manages secure local storage of credentials using platform key stores.
- Provides UI widgets for credential listing, detail viewing, presentation and revocation.
- Offers offline presentation using QR codes and NFC.
- **Usage:** Developers add the Dart package, initialise it with environment configuration (tenant ID, API base URL), and call `WalletSdk.issueCredential(...)` or `WalletSdk.presentCredential(...)` within the app. Full examples and integration tests are provided in the README.

.NET SDK (`/deliverables/sdk/dotnet`)

- **Purpose:** Enables agencies and backend services to integrate credential issuance and verification workflows in **C#** applications.
- **Features:**
- Client libraries for calling the issuance, status and revocation APIs.
- Support for OID4VCI pre-authorised flows, including JWT generation and signing.
- Verifier helper classes to validate presentations, check status lists and extract claims.
- Configuration helpers for multi-tenant environments and PKI trust configuration.
- **Usage:** Agencies reference the NuGet package, configure `WalletClientOptions` with client credentials and endpoints, and call methods such as `IssueCredentialAsync`,

`VerifyPresentationAsync` and `RevokeCredentialAsync`. The SDK integrates with ASP.NET identity providers and logs telemetry via built-in instrumentation.

Web/TypeScript SDK (`/deliverables/sdk/web`)

- **Purpose:** Allows web-based verifiers (e.g., relying parties, kiosks) to request and verify presentations.
- **Features:**
- Implements the OIDC4VP client to generate presentation requests and handle callbacks.
- Handles JSON-LD or JWT based verifiable presentations and validates signatures using the trust registry.
- Provides UI components (React hooks) for scanning QR codes and displaying verification results.
- **Usage:** Web developers install the npm package, instantiate a `VerifierClient` with configuration, call `createPresentationRequest()` to generate a QR code and `verifyPresentation()` when the citizen responds.

Versioning & deprecation policy

We follow **semantic versioning** (MAJOR.MINOR.PATCH) for all APIs and SDKs. Breaking changes increment the MAJOR version and are announced with at least six months' notice. The OpenAPI specification is versioned under `/api/v{n}`; previous versions remain available during the deprecation window. SDKs support the two most recent MAJOR releases.

Deprecation notices are communicated via release notes, DGov's change control board and our public developer portal. For example, if `v1` of the issuance API is superseded by `v2`, we will continue supporting `v1` for 12 months while encouraging migration. Deprecated endpoints emit warning headers to aid discovery.

This appendix summarises how the public APIs and SDKs are designed to meet the standards in Schedule 3 (specifically TS-7 and TS-8) and deliver a consistent developer experience across platforms.
