Australian Customer Data Standard - FAPI Security Profile

prepared by Francis Pouatcha for SFTI

adorsys

Australian CDS FAPI Profile

Overview

Standards

oAuth:

- IETF - RFC6749 The OAuth 2.0 Authorization Framework
- Focus on authorization flows (a.k.a authZ)
- Has many BCPs and complementary RFCs

OIDC:

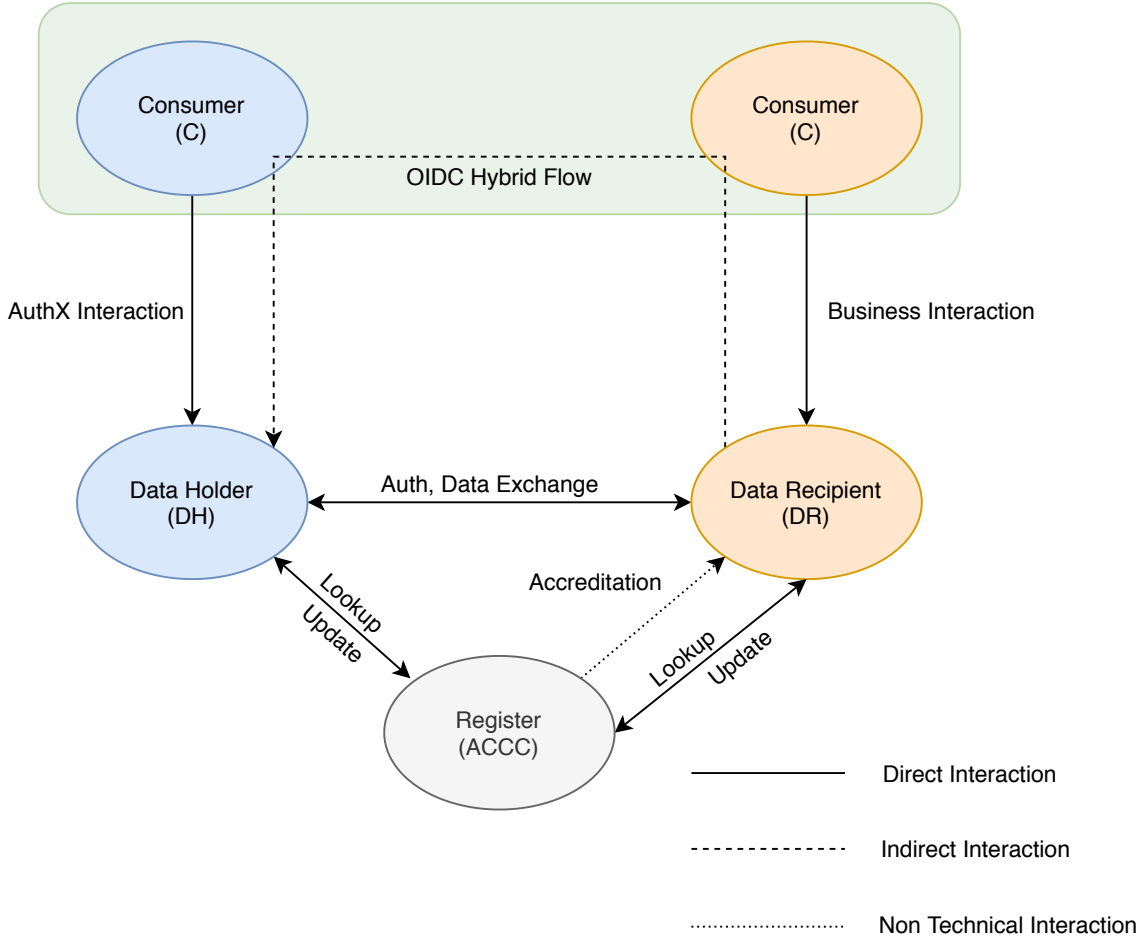
- OpenID Connect standard (https://openid.net/connect/)
- https://openid.net/specs/openid-connect-core-1_0.html
- Extends oAuth with authentication requirements (a.k.a authN)
- Has many flow combinations

FAPI:

- OIDC Profiles for financial APIs
- Limits flows and encryption algorithms to allow for secure application of OIDC to financial industry
- Defines a READ (R) and a READ-WRITE (RW) profile
- Extends oAuth/OIDC with new drafts (JARM, PAR, JAR, RAR, ..)

CDS Security Profile:

- Materializes FAPI for Australian CDS
- Limit AuthZ flow to **OIDC hybrid flow with confidential clients**
- Mandates a second factor for user authN
- Defines a governing framework with an accreditation process and a register for accredited Data Recipients (DR).
- Register is run by ACCC (**A**ustralian **C**ompetition and **C**onsumer **C**ommission)
- Register provides SSL Certificates for Data Holders and accredited Data



Authorization Flow

Customer Present Interfaces

oAuth:

- At Auth-Endpoint: **response_type** is XOR of [code**,** token]
- Implicit Flow: token
- Authorization Code Flow: code
- Token-Endpoint: **grant_type** is XOR of [authorization_code, password, client_credentials, refresh_token]

OIDC:

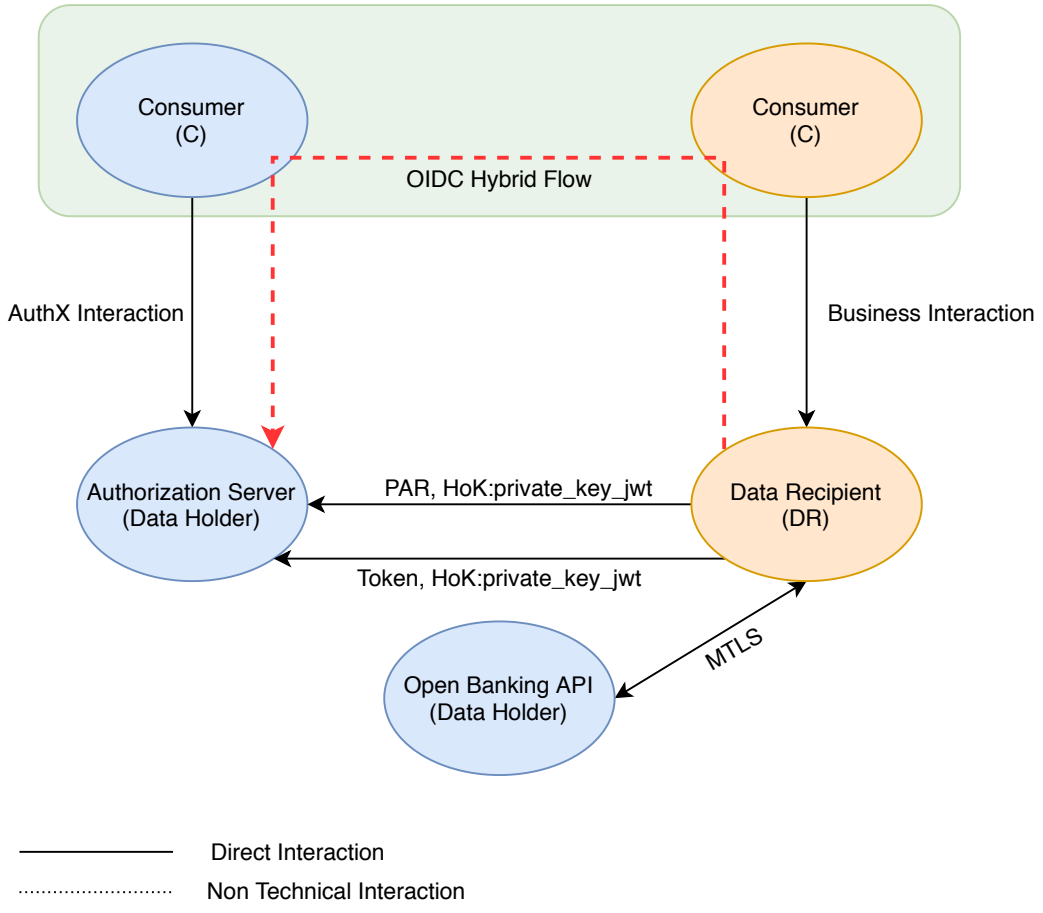
- **scope: openid**
- At Auth-Endpoint: response_type is OR of [code**,** token, **id_token**]
- response_type controls authN (when and where to issue id_token)
- Front-channel only: r_t=token, r_t=id_token, r_t=id_token token
- Back-channel is code: r_t=code ... eg. r_t=code token
- Even mor complexity: r_t=code token id_token ???

FAPI (RW):

- Narrows down OIDC for use with financial APIs
- **Mandates** OIDC params request or request_uri (signed and optionaly encrypted) -> see: https://tools.ietf.org/html/draft-ietf-oauth-jwsreq-30
- Limits flow: r_t=code id_token, r_t=code & r_m=jwt (JARM)
- sender constrain token: PoP!, allow MTLS for sender PoP
- allows PAR, introduces PKCE

CDS Security Profile:

- Materializes FAPI for Australian CDS
- Limit client types to **confidential** (no public clients)
- Allows only: r_t=code id_token (eliminating complexity)
- Precision: request_uri only if PAR is supported
- Customer identifier per Data Holder (no correlation)
- No password at Data Recipient interface (no embedded like BG)
- Mandatory second factor (OTP)
- MTLS https://tools.ietf.org/html/rfc8705 (but no PKI based MTLS), accept only certificates issued by CDR Certificate Authority



Australian CDS FAPI Profile

Back Channel Authentication

SWISS FINTECH INNOVATIONS

Auth Server Back Channel Endpoints

PAR:

- Pushed Access Request (seehttps://tools.ietf.org/html/draft-lodderstedt-oauth-par-01)
- Optional for CDS
- Use a back channel endpoint to send authZ details to IDP

Token:

- oAuth2 token endpoint
- private_key_jwt must be sent as part of the body part
- private_key_jwt must contain other parameter required at token endpoint (e.g.: grant_type, code, clien_id, ...)

Transaction Endpoints

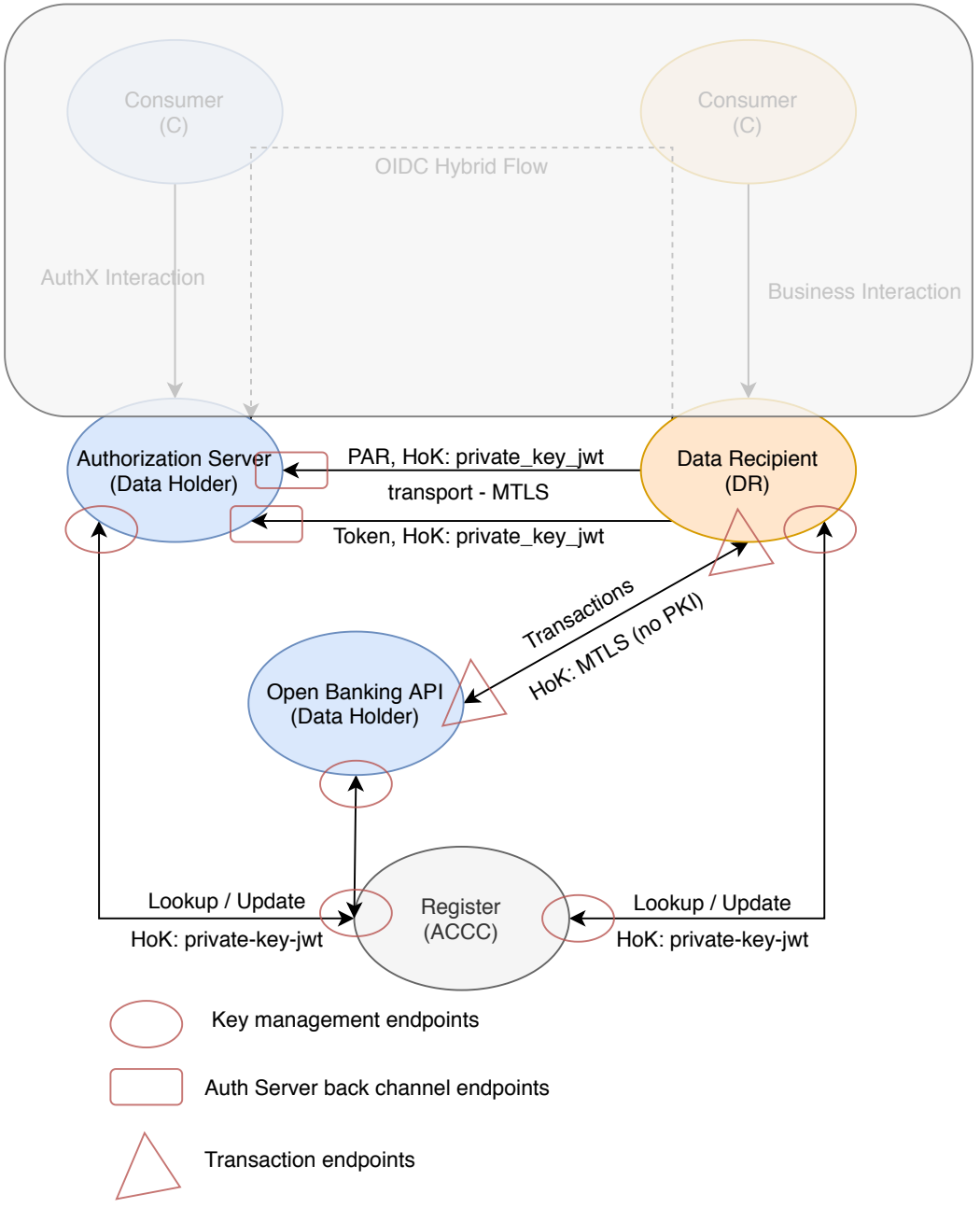
- In both direction DH2DR and DR2DH, use MTLS to protect transactions. MTLS (without PKI) see https://tools.ietf.org/html/rfc8705
- Use MTLS for transport and token binding a.k.a Holder of Key.
- This does not provide non repudiation. Non repudiation requires request signature!

Key Management Endpoints

- All key management endpoint follow the same principle
- Caller authenticate with signed JWT. Very closed to the private_key_jwt
- Register maintains a directory for all DH and accredited DR
- Digital certificate associated with DH/DR can be looked up at Register
- Register Key is known to all!

JWKS Endpoints

- Exposes participants public keys (signature keys, encryption keys)
- Data Holder public keys available at OIDC end point registered with CDR Register
- Data Recipient public keys available at the URI registered with CDR Register
- CDR Register public keys available at the CDR Register end point exposed for that purpose



adorsys

SWISS FINTECH INNOVATIONS

Data Holders calling Data Recipients

Data Recipients MUST support the authentication of Data Holders using a signed JWT according to the following requirements:

- The JWT MUST contain the following REQUIRED Claim Values and MAY contain the following OPTIONAL Claim Values:
  - `iss` - REQUIRED. Issuer. This MUST contain the id of the Data Holder obtained from the CDR Register.
  - `sub` - REQUIRED. Subject. This MUST contain the id of the Data Holder obtained from the CDR Register.
  - `aud` - REQUIRED. Audience. The aud (audience) Claim. Value that identifies the Data Recipient as the intended audience. The Data Recipient MUST verify that it is an intended audience for the token. Contents MUST be the base URI for the end point being accessed.
  - `jti` - REQUIRED. JWT ID. A unique identifier for the token, which can be used to prevent reuse of the token. These tokens MUST only be used once.
  - `exp` - RE QUIRED. Expiration time on or after which the ID Token MUST NOT be accepted for processing.
  - `iat` - OPTIONAL. Time at which the JWT was issued.
- Validation and use of the JWT and the claims described above MUST be performed in accordance with [JWT]
- The JWT should be accepted from the client using the "Authorization Request Header Field" mechanism as described in section 2.1 of RFC6750

adorsys

SWISS FINTECH INNOVATIONS

Data Recipients calling Data Holders

Data Holders MUST support the authentication of Data Recipients using the `private_key_jwt` Client Authentication method specified at section 9 of **[OIDC]**.

The `private_key_jwt` authentication method is enabled through the delivery of an encoded **[JWT]** signed using the Data Recipient's private key and thus facilitates non-repudiation.

Client public keys are obtained from the **[JWKS]** endpoints.

The **[JWT]** represents an assertion that MUST include the following claims:

- `iss`: The client ID of the bearer.
- `sub`: The client ID of the bearer.
- `aud`: The Token Endpoint URL.
- `exp`: A JSON number representing the number of seconds from 1970-01-01T00:00:00Z to the UTC expiry time.
- `jti`: A unique identifier generated by the client for this authentication.

The following claims MAY be included:

- `iat`: A JSON number representing the number of seconds from 1970-01-01T00:00:00Z to the UTC issued at time.

Front Channel Authentication

SWISS FINTECH INNOVATIONS

Customer Present Interfaces

Authorization Request:

- scope: openid
- response_type=code id_token

ID Token:

- Returned in front channel with authorization_code, in back channel with access token.
- Must be signed and encrypted (DR keys from DR's jwks-url)
- sub (subject)
- acr
- auth_time
- name, given_name, family_name, updated_at
- **Not Personal Information (PI)** when ID Token returned from front channel endpoints.
- ID Token secure further response values: c_hash (code), s_hash (state)

Access Token:

- Delivery container object contains all information listed bellow
- Expires: 2 minutes < expires_in < 10 minutes
- Sender constrained (Holder of Key mechanism - DR ID)
- Refreshable (refresh_token_expires_at)
- Consent Management: sharing_expires_at (sharing_duration), cdr_arrangement_id

Refresh Token:

- Mandatory
- 28 Days < refresh_token_expires_at < sharing_expires_at
- Different from agreement expiry -> sharing_expires_at
- Can be recycled with each access token request
- Can be opaque to client

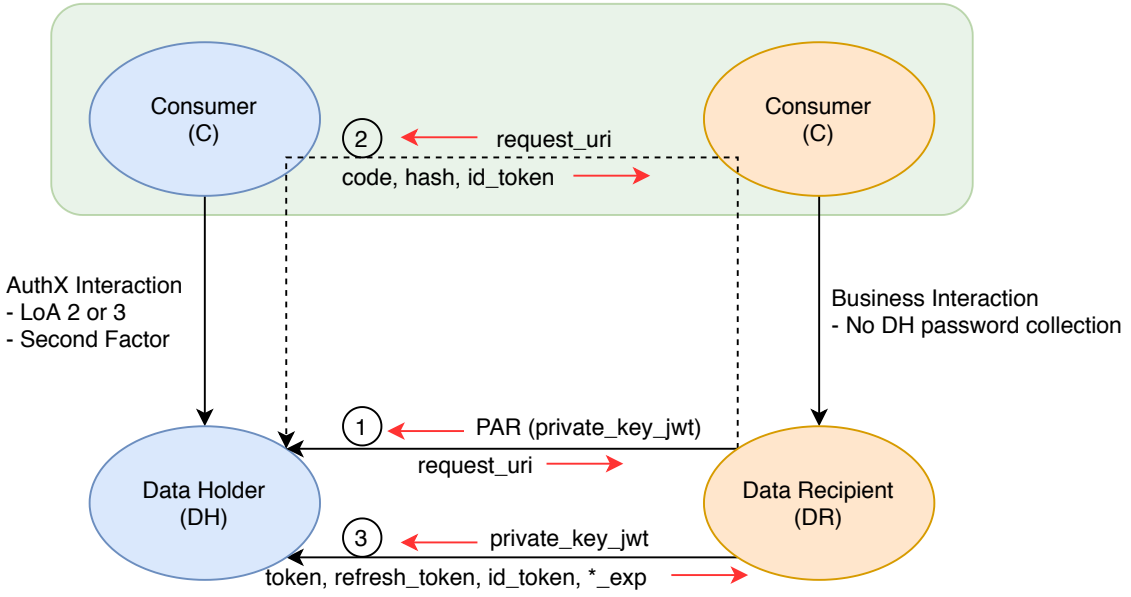
Customer Present Interfaces

CDR Arrangement ID:

- Consent arrangement between a DR and DH for a given consumer (NextGenPSD2 consent-id)
- Unique in context of DH, non guessable
- No correlation across DH
- Essential for management of consent life cycle
- Field name: cdr_arrangement_id, mandatory from Nov. 2020

Levels of Assurance (LoA):

- Required
- Use claim acr (Authentication Context Class Reference)
- Built on TDIF (Australian Trusted Digital Identity Framework)
- Read Operations: LoA 2 --> TDIF-CL1
- Write Operations: LoA 3 --> TDIF-CL2
- Consent establishment requires LoA 2.



adorsys

OpenID Provider Configuration

Data Holders MUST make their OpenID Provider Metadata available via a configuration end point as outlined in Section 3 and 4 of the OpenID Connect Discovery standards **[OIDD]**.

At a minimum, the Data Holder metadata MUST include:

- `issuer`: URL that the Data Holder asserts as its Issuer Identifier.
- `authorization_endpoint`: URL of the Authorization End Point.
- `token_endpoint`: URL of the Token End Point.
- `introspection_endpoint`: URL of the Introspection End Point.
- `revocation_endpoint`: URL of the Revocation End Point.
- `userinfo_endpoint`: URL of the UserInfo End Point.
- `registration_endpoint`: URL of the Client Registration End Point.
- `scopes_supported`: This list of supported scopes.
- `claims_supported`: The list of supported claims.
- `acr_values_supported`: The supported ACR values.
- `jwks_uri`: The JSON Web Key Set for the data holder.
- `id_token_encryption_alg_values_supported`: The list of the supported JWE algorithms for securing the issued ID tokens. Must conform to **[FAPI-RW]** and **[OIDD]**.
- `id_token_encryption_enc_values_supported`: The list of the supported JWE encryption methods for securing the issued ID tokens.

Questions?

adorsys