April 9, 2020

Data Standards Body
Data61
5/13 Garden St
Eveleigh NSW 2015
Australia

RE: Decision Proposal 99 - Concurrent Consent

The Financial-Grade API Working Group (FAPI WG) thanks the Australian Data Standards Body for the opportunity to respond to *Decision Proposal 099 - Concurrent Consent (DP99)*[1] published by the Data Standards Body published on 26th of March 2020. As previously communicated we welcome the opportunity to participate in an open collaboration that leverages the combined experiences of the FAPI WG to deliver a world leading open data outcome as part of the Australian Consumer Data Right (CDR) and the Consumer Data Standards (CDS).

In order to provide concise feedback on the proposal we provide comment on a number of key topics:

1. FAPI WG & Data61 Proposal Comparison
2. Adoption of the FAPI 2.0 profile
3. Adoption of Pushed Authorisation Request (PAR)
4. Adoption of JWT Secured Authorization Request
5. Non adoption of Rich Authorisation Request (RAR)
6. Non Adoption of Grant Management Extension & API
7. Sharing Agreement API Proposal
8. Pathway for adoption in the context of the existing Consumer Data Standards

---

1 https://github.com/ConsumerDataStandardsAustralia/standards/files/4384751/Decision.Proposal.99.-.Concurrent.Consent.pdf

FAPI WG & Data61 Proposal Comparison

#	Feature	FAPI WG Target State proposal https://github.com/ConsumerDataStandardsAustralia/standards/issues/99#issuecomment-592320557	New decision proposal Change proposed by Data61 ("November" Release) https://github.com/ConsumerDataStandardsAustralia/standards/issues/99#issuecomment-604196943	FAPI WG New proposal for Stage 1 ("November" Release)
1	Consent identifier to support concurrent consent	✔️Grant Management's grant_id as **OAuth parameter**	☝️New **sharing_id OIDC** claim	✔️Adopt Grant Management's grant_id as **OAuth parameter** Grant Management already defines grant_id, how it should be issued and used. Vendors supporting Australian Data Holders are actively contributing to the specification. sharing_id is custom and limited to OIDC use cases. 'Grant' works better for other use cases (e.g. 'write' and etc).
2	Backchannel Request lodgement	✔️Adoption of PAR	✔️Adoption of PAR	☝️Defer PAR adoption till Stage 2* (for fine-grained consent) - continue using existing signed Request Objects in Authorisation Request until Stage 2. PAR is still in OIDF target state.
3	Rich (fine-grained) Authorisation Request	✔️Adoption of RAR	❌Non Adoption of RAR	☝️Adopt RAR but use simple RAR until Stage 2 (for fine-grained consent). Simple RAR object to be expanded in the later release minimises the amount of change on the data holders and data recipients in the future.
4	Signed and Encrypted JWT request	✔️JAR is not required (Covered by PAR+RAR)	☝️Adoption of JAR	❌Don't adopt JAR for Stage 1 - continue using existing signed Request Object in Authorisation Request until Stage 2* (when PAR is adopted). ✔️Adopt JAR in Stage 2 if non-repudiation is required after PAR is introduced or standard

				signed request object is required.
5	“Consent API”	✔️Data Holder: Grant Management API (GET and DELETE)	❌Not adopting Grant Management API ☝️New Sharing Agreement Management API for revocation (DELETE only)	☝️Defer Grant Management API (GET & DELETE) until Stage 2*. - GET - when multi-party consent is required or it’s used for consent change / revocation notification). - DELETE - see #7 Proposed grant management API covers all functionality of Sharing Agreement Management API and more. By adopting this specification early, Australia can maximise the chance for future standards alignment, vendor support and leverage expertise of OIDF and its community. Grant Management API is still in OIDF target state.
6	Decoupled re-authorisation request	✔️Adopt CIBA	☝️Defer CIBA adoption till later	☝️Defer CIBA adoption until Stage 2*+ (when re-authorisation is required). CIBA is still in OIDF target state.
7	**Data Holder**’s consent revocation API	✔️Grant Management API (DELETE)	☝️New Sharing Agreement Management API for revocation (DELETE only)	☝️Continue using Token Revocation endpoint with intent to migrate to Grant Management API DELETE (Stage 2*).
8	**Data Recipient’s** consent revocation notification by Data Holder	N/A (not covered)	☝️Data Recipient: Consent revocation by Data Holder is done using New Sharing Agreement Management API for revocation (DELETE only)	☝️Find alternative way to notify Data Recipient of consent revocation without complicating the ecosystem (multi-directional communication) and imposing API hosting, availability and client authentication requirements on Data Recipients and additional complexity on Data Holders.

* *Stage 2 elements can be designated as OPTIONAL (“MAY”) elements within earlier specifications.*

Adoption of the FAPI 2.0 Profile

The FAPI WG welcomes the Data Standards Body's steps towards adoption of the FAPI 2.0 Baseline profile. As the leading solution for high security and enhanced integrity of data transfer FAPI 2.0 combines a significant number of lessons learned in numerous jurisdictions over more than 5 years, a well defined attacker model[2], a deep analysis[3] of potential attack scenarios as well as a feature rich framework for building security sensitive applications based on multiple established or emerging international standards including Pushed Authorization Request (PAR), Rich Authorization Requests (RAR) and Proof Key for Code Exchange (PKCE).

With respect to DP99 we wish to make the following observations:

- PAR is mandated within this profile which, when coupled with Mutual TLS for client authentication, may remove the necessity for armoured (ie. signed) Request Objects being required. This could provide a measurable improvement in ease of implementation for Data Recipients.
- PKCE is mandated within the profile as the code challenge method. This eliminates the requirement for front channel ID Tokens to be exchanged resulting in:
    - Removal of ID Token exchange in front channel simplifying the developer experience
    - ID Tokens which convey Personally Identifiable Information (PII) can continue to be passed via the backchannel removing the requirement for encrypted ID Tokens to be used as is currently the case within CDS
    - Simplification for user agent initiation by allowing for *HTTP GET* calls to the */authorize* endpoint without potential exposure to query string length limits
- Resource Servers are mandated to verify access, including permissions conveyed within RAR based *authorization_details*

Adoption of Pushed Authorisation Request (PAR)

The FAPI WG is encouraged by the DSB's proposal to adopt PAR as part of DP99. With 5 known implementations (but still formally a OAuth Working Group Draft) PAR represents a simple solution with respect to large requests being dispatched via the front channel.

---

[2] https://bitbucket.org/openid/fapi/src/master/FAPI_2_0_Attacker_Model.md

[3] https://www.google.com/url?q=https://docs.google.com/spreadsheets/d/1PtG4f-Svils7wHBa7cGaZubbh-6lGifce38c_oShSss/edit?usp%3Dsharing&sa=D&ust=1586354639378000&usg=AFQjCNG7-lXEUj6FLPRVGMajGliETO8Wmg

Adoption of PAR results in a significant reduction in complexity for Data Recipients with regards to application design during user-agent authorisation initiation. Further, PAR requires client authentication facilitating enhanced Data Recipient verification by Data Holders, increasing the security of the overall flow.

On a related note this specification is mandatory for FAPI 2.0 compliance and consequently its adoption progresses the CDR towards adoption of the next evolution of this highly secure profile.

Finally, the examples provided within DP99 include mixed content parameters. PAR does not support mixed content parameters and consequently the example supplied should include either a reference *or* body parameters (ie. client_id)

Adoption of JWT Secured Authorization Request (JAR)

The FAPI WG supports the decision to adopt JAR long term but doesn't believe it is required for Stage 1 (November release) if Data61 needs to prioritise standards to be adopted.

Non-adoption of Rich Authorization Request (RAR)

The FAPI WG notes that while RAR was considered within DP99 it was not adopted on the basis that it is an *"emerging draft"* and that the decision proposal does not seek to address fine-grained authorisation.

The FAPI WG wishes to highlight that while the Rich Authorization Request specification is formally in *Draft* status, the specification itself is now in its 3rd iteration[4] and is considered by its authors to be stable with no significant changes in over 6 months. This is a similar status to the current status of DSB adopted specifications of PAR[5] and JAR[6] which all have overlapping authors.

Within the context of DP99 and the broader Consumer Data Standards, FAPI WG members feel that there is significant additional value gained through its immediate adoption when considered in the context of the current approach of adding "processed" claims to the root of the underlying request object. This includes:

- Dynamic definition of jurisdictionally specific requirements within a defined framework (*authorization_details*)
- Centralisation of authorization details into a single location for interrogation by both Operating Parties (for Resource Server introspection) and Relying Parties (for permission discovery)
- Support inherited through the FAPI 2.0 profile

[4] https://tools.ietf.org/html/draft-ietf-oauth-rar
[5] https://tools.ietf.org/html/draft-ietf-oauth-par
[6] https://tools.ietf.org/html/draft-ietf-oauth-jwsreq-20

- Strong enrichment support through subsequent versions which can be defined, at the DSBs discretion, in future versions of authorization requests using the specification-defined *type* field
- Future-proofing for what are already clear future requirements including:
    - Joint Account support including for progressive Consent Lifecycle
    - Fine Grained Authorisation via multiple dimensions dependent on industry specific (ie. energy, telco etc) requirements

Given its stability and emerging use cases currently being implemented by working group participants the working group strongly feels that the adoption of RAR is of no larger complexity, and indeed due to global vendor support quite possibly less complexity, than the currently proposed approach taken by the DSB with respect to the use of claims which convey “business process” constraints as evidenced by the use of *sharing_duration*.

To further articulate how RAR could be incorporated into the Consumer Data Standards the working group includes sample payloads which seek to align with the current requirements the Consumer Data Standards currently specify.

Sharing Agreement API Proposal

The FAPI WG read with interest the proposal for a Sharing Agreement API within DP99. This proposal appears to be primarily targeted as an alternative to the Revocation of Consent method defined in the existing Standards.

The FAPI Working Group wishes to highlight that the proposed Sharing Agreement API is very similar to the UK Open Banking’s (OBIE) Web Hook Revocation method. In addition to this method OBIE also makes available a Bulk Consent Status API and the Lodged Intent Consent Management API.

Observationally an overwhelming majority of participants prefer the Lodged Intent Consent Management API. The primary reason for this is driven by the multitude of failure scenarios introduced through the requirement for reliable bi-directional communication between the TPP (Holder) and ASPSP (Recipient).

Non-Adoption of Grant Management Extension & API

The Decision Proposal states that the Grant Management Extension & API is an *emerging draft* and ostensibly this appears to be the reason for limited consideration at this stage. While the specification itself is nascent, it is the culmination of three years of experiences across multiple jurisdictions including UK Open Banking, Polish, Czech and other PSD2 governed jurisdictions.

Specifically related to the UK Open Banking ecosystem (which CDR was intended to be guided

by) the Grant Management API incorporates a large number of improvements to resolve shortfalls of Intent Lodgement pattern and anti-patterns observed within the UK ecosystem.

Finally, the working group wishes to highlight that notwithstanding evolution of the Grant Management API the same holds true for any CDR specific solution with the key difference being that a CDR specific solution would evolve without the support of the broad community of the OpenID Foundation (OIDF).

The Data Standards Body is welcome, and indeed encouraged, to contribute to the evolution of the Grant Management API specification.

FAPI WG members also wish to highlight that the discovery of consent status can be solved, in a more reliable and architecturally simplified way through the use of the Grant Management API's query functionality.

Grant Management vs. Sharing Agreement

In addition to this existing ecosystem observation the FAPI WG wishes to note the following:

- The proposed *sharing_id* is not a universal consent identifier as it appears to require OpenID Connect to function. Conversely, *grant_id* is an OAuth2 extension which can be uniformly used by all OAuth2 based applications
- The mechanics for the Sharing Agreement AI assumes that the OP is authoritative for consent decisions with respect to disclosure. The Grant Management API enhances interoperability by providing capability for the RP (Data Recipient) to explicitly request components they wish to request exposure on
- The Sharing Agreement API does not consider use cases beyond sharing limiting its utility as the CDR expands to other capabilities
- The Sharing Agreement API appears to attempt to abstract live authorisations (ie. "consents") away from the authorisation server itself. This has flow-on impacts when considered in the context of data security domains.
- The Grant Management API supports requesting of grant status which significantly reduces the complexity required for notification of consent revocations
- The Grant Management API has an established security baseline, is authored by those with background experience in active deployments and has registered interest from established vendors/implementers of industry leading software toolsets

Pathway for adoption in the context of the existing Consumer Data Standards

The FAPI Working Group understands that the DSB wishes to deliver a solution for Concurrent Consent by November 2020. Not-withstanding the viability of this timeline with respect to the existing Decision Proposal, the Working Group believes that the most prudent course of action towards an eventual FAPI 2.0 compliant adoption while considering communicated delivery timelines would be as follows:

1. Stage 1 (Currently November 2020):
    a. Adopt Rich Authorisation Request and migrate existing custom claims into a *cdr_sharing_v1* authorization_details. Retain existing Request Object signing as currently defined within the Consumer Data Standards.
    b. Adopt initial grant_management_mode of *create* facilitating the delivery of concurrent consents and Grant identifiers
    c. Retain the existing Token Revocation call between Data Holder and Data Recipient
2. Stage 2:
    a. Adopt Grant Management API with *query* capability to allow for Data Recipients to regularly poll for Consent Status and therefore detect revocation events removing the need for bi-directional communication
    b. Decommission the Token Revocation requirement between Data Holder and Data Recipient
    c. Convert to S256 code challenge method thereby removing;
        i. ID Token exchange on Front Channel
        ii. Requirement for Signed and Encrypted ID Tokens
3. Stage 3:
    a. Transition the existing Request Object to be loaded via Pushed Authorisation Request
    b. Introduce grant management API modes of *revoke* to enable Recipients to revoke individual grants without forcibly terminating a session
    c. Introduce Grant Management Modes of *update* and *replace* to facilitate inline consent upgrade/downgrade capability

Appendix 1: Sample Payloads

Stage 1

Initial adoption of RAR within existing Request Object structure with a request sharing duration of 90 days, a request for *sharing_expires_at* and *sharing_status* to be returned.

Authorise Request and Request Object JWT

```
GET /authorise?
 response_type=code%20id_token
   &client_id=12345
   &redirect_uri=https%3A%2F%2Fwww.recipient.com.au%2Fcoolstuff
   &scope=openid%20profile
   &nonce=n-0S6_WzA2Mj
   &grant_management_mode=create
   &state=af0ifjsldkj
   &request=eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjEyMyJ9.ey ...
Host: www.holder.com.au

Decoded Request JWT
{
        "alg": "PS256",
        "typ": "JWT",
        "kid": "123"
}
{
        "aud": "https://www.recipient.com.au",
        "response_type": "code id_token",
        "client_id": "12345",
        "redirect_uri": "https://www.recipient.com.au/coolstuff",
        "scope": "openid",
        "state": "af0ifjsldkj",
        "nonce": "n-0S6_WzA2Mj",
        "authorization_details": [
            {
                "type": "cdr_sharing_v1",
                "actions":[bank:accounts:basic:read],
                "sharing_duration": 7776000,
                "sharing_expires_at": ""
            }
        ]
        "claims": {
            …
        }
    }
}
```

Token Response

```
POST /token HTTP/1.1
Host: www.holder.com.au
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&
  code=i1WsRn1uB1&
  client_id=s6BhdRkqt3&
  client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer&
  client_assertion=eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjEyNDU2In0.ey ...

HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-cache, no-store

{
      {
       "access_token": "2YotnFZFEjr1zCsicMWpAA……..",
       "token_type": "bearer",
       "expires_in": 3600,
       "refresh_token": "tGzv3JOkF0XG5Qx2TlKWIA…..",
       "refresh_token_expires_at": "1311281970",
       "authorization_details": [
           {
                "type": "cdr_sharing_v1",
                "sharing_duration": 7776000,
                "actions":["bank:accounts:basic:read"],
                "sharing_expires_at": "1311281970"
           }
        ]
       "grant_id":"TSdqirmAxDa0_-DB_1bASQ"
    }
}

# Decoded JWT

{
      "iss": "https://www.holder.com.au",
      "sub": "a9ebbef6-1f0b-44eb-96cf-0c5b51b37ab2",
      "aud": "a7AfcPcsl2",
      "exp": 1311281970,
      …
      "scope": "openid bank:accounts.basic:read", # Maintained for compatibility
      ...
}
```

Stage 2

Adopt Grant Management *query* API and convert to PKCE S256 code challenge mechanism.

Stage 2 - Authorise Request and Request Object JWT with PKCE

```
GET /authorise?
 response_type=code
   &client_id=12345
   &redirect_uri=https%3A%2F%2Fwww.recipient.com.au%2Fcoolstuff
   &scope=openid%20profile
   &code_challenge=af0ifjsldkj
   &code_challenge_method=S256
   &grant_management_mode=create
   &request=eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjEyMyJ9.ey ...

Host: www.holder.com.au

Decoded Request JWT
{
        "alg": "PS256",
        "typ": "JWT",
        "kid": "123"
}
{
        "aud": "https://www.recipient.com.au",
        "response_type": "code id_token",
        "client_id": "12345",
        "redirect_uri": "https://www.recipient.com.au/coolstuff",
        "scope": "openid",
        "state": "af0ifjsldkj",
        "nonce": "n-0S6_WzA2Mj",
        "authorization_details": [
            {
                "type": "cdr_sharing_v1",
                "“actions”:[bank:accounts:basic:read],
                "“sharing_duration”: 7776000,
                "“sharing_expires_at”: “”,
                "“sharing_status”: “”
            }
        ]
        "claims": {
            …
        }
    }
}
```

Stage 2 - Token Request and Response with introduction of *sharing_status*

```
POST /token HTTP/1.1
Host: www.holder.com.au
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&
  code=i1WsRn1uB1&
  code_verifier=iWsBrn1uBR
  client_id=s6BhdRkqt3&

HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-cache, no-store

{
      {
       "access_token": "2YotnFZFEjr1zCsicMWpAA……..",
       "token_type": "bearer",
       "expires_in": 3600,
       "refresh_token": "tGzv3JOkF0XG5Qx2TlKWIA…..",
       "refresh_token_expires_at": "1311281970",
       "grant_id":"TSdqirmAxDa0_-DB_1bASQ"
    }
}

# Decoded JWT

{
      "iss": "https://www.holder.com.au",
      "sub": "a9ebbef6-1f0b-44eb-96cf-0c5b51b37ab2",
      "aud": "a7AfcPcsl2",
      "exp": 1311281970,
      …
      "scope": "openid bank:accounts.basic:read", # Maintained for compatibility
      "authorization_details": [
           {
                "type": "cdr_sharing_v1",
                "sharing_duration": 7776000,
                "actions":["bank:accounts:basic:read"],
                "sharing_expires_at": "1311281970"
                "sharing_status": "ACTIVE"
           }
        ],
      ...
}
```

Stage 2 - Authorization Details with Grant Management GET (ACTIVE)

```
GET /grants/TSdqirmAxDa0_-DB_1bASQ
Host: as.example.com
Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA

HTTP/1.1 200 OK
Cache-Control: no-cache, no-store
Content-Type: application/json

{
  "authorization_details": [
       {
            "type": "cdr_sharing_v1",
            "sharing_duration": 7776000,
            "actions":["bank:accounts:basic:read"],
            "sharing_expires_at": "1311281970"
            "sharing_status": "ACTIVE"
       }
    ]
}
```

Stage 2 - Authorization Details with Grant Management GET (REVOKED)

```
GET /grants/TSdqirmAxDa0_-DB_1bASQ
Host: as.example.com
Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA

HTTP/1.1 200 OK
Cache-Control: no-cache, no-store
Content-Type: application/json

{
    "authorization_details": [
       {
            "type": "cdr_sharing_v1",
            "sharing_duration": 7776000,
            "actions":["bank:accounts:basic:read"],
            "sharing_expires_at": "1586353683", # "now"
            "sharing_status": "REVOKED"
       }
    ]
}
```

Stage 3

Move to Pushed Request Object, introduce Grant Management API Revoke. All body parameters are within a Signed Request Object.

Stage 3 - PAR with RAR and PKCE (FAPI 2 style)

```
POST /as/par
HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded

request=eyJhbGciOiJI1sInR5cCI6IkpXVCIsImtpZCI6IjEyMyJ9.ey

# Decoded JWT
{
      "iss": "https://www.holder.com.au",
      "aud": "a7AfcPcsl2",
      "exp": 1311281970,
      “client_id”: s6BhdRkqt3,
      “code_challenge”: af0ifjsldkj,
      “code_challenge_method”: S256,
      “grant_management_mode”: create,
      "authorization_details": [
           {
                "type": "cdr_sharing_v1",
                “sharing_duration”: 7776000,
                “actions”:[“bank:accounts:basic:read”],
                "sharing_expires_at": "1311281970"
                “sharing_status”: “ACTIVE”
           }
        ],
      ...
}
HTTP/1.1 201
Created Cache-Control: no-cache, no-store
Content-Type: application/json
{
   "request_uri": "urn:example:bwc4JK-ESC0w8acc191e-Y1LTC2",
   "expires_in": 90
}
```

Stage 3 - Authorise Request and Request Object JWT

```
GET /authorize?
    request_uri= urn%3Aexample%3Abwc4JK-ESC0w8acc191e-Y1C2

Host: www.holder.com.au
HTTP/1.1 200 OK
```

Stage 3 - Token Request and Response

```
POST /token HTTP/1.1
Host: www.holder.com.au
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&
  code=i1WsRn1uB1&
  code_verifier=iWsBrn1uBR
  client_id=s6BhdRkqt3&

HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-cache, no-store
{
      {
       "access_token": "2YotnFZFEjr1zCsicMWpAA……..",
       "token_type": "bearer",
       "expires_in": 3600,
       "refresh_token": "tGzv3JOkF0XG5Qx2TlKWIA…..",
       "refresh_token_expires_at": "1311281970",
       "grant_id":"TSdqirmAxDa0_-DB_1bASQ",
       "authorization_details": [
           {
                "type": "cdr_sharing_v1",
                "sharing_duration": 7776000,
                "actions":["bank:accounts:basic:read"],
                "sharing_expires_at": "1311281970"
                "sharing_status": "ACTIVE"
           }
       ],
    }
}

# Decoded JWT
{
      "iss": "https://www.holder.com.au",
      "sub": "a9ebbef6-1f0b-44eb-96cf-0c5b51b37ab2",
      "aud": "a7AfcPcsl2",
      "exp": 1311281970,
      ...
}
```

Stage 3 - Authorization Details with Grant Management GET (ACTIVE)

```
GET /grants/TSdqirmAxDa0_-DB_1bASQ
Host: as.example.com
Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA

HTTP/1.1 200 OK
Cache-Control: no-cache, no-store
Content-Type: application/json
{
  "authorization_details": [
       {
            "type": "cdr_sharing_v1",
            "sharing_duration": 7776000,
            "actions":["bank:accounts:basic:read"],
            "sharing_expires_at": "1311281970"
            "sharing_status": "ACTIVE"
       }
    ]
}
```

Stage 3 - Grant Management Revoke

```
DELETE /grants/TSdqirmAxDa0_-DB_1bASQ
Host: as.example.com
Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA

HTTP/1.1 201 OK
Cache-Control: no-cache, no-store
Content-Type: application/json
```

Stage 3 - Authorization Details with Grant Management GET (REVOKED status)

```
GET /grants/TSdqirmAxDa0_-DB_1bASQ
Host: as.example.com
Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA

HTTP/1.1 200 OK
Cache-Control: no-cache, no-store
Content-Type: application/json
{
    "authorization_details": [
       {
            "type": "cdr_sharing_v1",
            "sharing_duration": 7776000,
            "actions":["bank:accounts:basic:read"],
            "sharing_expires_at": "1586353683", # "now"
            "sharing_status": "REVOKED"
       }
    ]
}
```

References

Specification	Description	Status
FAPI 1 - Read profile (Part 1) https://openid.net/specs/openid-financial-api-part-1-ID2.html	Part 1 of Financial-grade API is a profile of OAuth that is suitable to be used in the access of read-only financial data and similar use cases.	Adopted by Australian CDR and Open Banking UK.
FAPI 1 - Read / Write profile (Part 2) https://openid.net/specs/openid-financial-api-part-2-ID2.html	Part 2 of Financial-grade API is a profile of OAuth that is suitable to be used in write access to financial data (also known as transaction access) and other similar higher risk access.	Adopted by Australian CDR and Open Banking UK.
PAR - Pushed Authorization Request https://tools.ietf.org/html/draft-ietf-oauth-par	Providing an interoperable way to push the payload of a request object directly to the AS in exchange for a "request_uri".	Active IETF OAuth WG draft (stable)
RAR - Rich Authorization Request https://tools.ietf.org/html/draft-ietf-oauth-rar	New OAuth parameter "authorization_details" that allows clients to specify their fine-grained authorization requirements using the expressiveness of JSON data structures.	Active IETF OAuth WG draft Base RAR support can be done before vendors fully support RAR
Grant Management for OAuth 2.0 https://bitbucket.org/openid/fapi/src/master/Financial_API_Grant_Management.md	OAuth extension to expose grant_id and APIs to query the status of and revoke grants.	Active FAPI WG draft
FAPI CIBA - Client Initiated Backchannel Authentication https://openid.net/specs/openid-financial-api-ciba-ID1.html	A profile of the OpenID Connect Client Initiated Backchannel Authentication Flow [CIBA] that supports this decoupled interaction method.	Implementer's Draft (stable)
JAR - JWT Secured Authorization Request https://tools.ietf.org/html/draft-ietf-oauth-jwsreq-20	The ability to send request parameters in a JSON Web Token (JWT) instead, which allows the request to be signed with JSON Web Signature (JWS) and encrypted with JSON Web Encryption (JWE) so that the integrity, source authentication and confidentiality property of the Authorization Request is attained.	Active IETF OAuth WG draft
FAPI 2.0 Baseline Profile https://bitbucket.org/openid/fapi/src/master/FAPI_2_0_Baseline_Profile.md **FAPI 2.0** Attacker Model https://bitbucket.org/openid/fapi/src/master/FAPI_2_0_Attacker_Model.md	Next Generation FAPI specification.	Active FAPI WG draft