Data Standards Body

Technical Working Group

Decision Proposal 99 – Finalisation of concurrent consent

Contact: Mark Verstege

Publish Date: 26/03/2020

Feedback Conclusion Date: 09/04/2020

Context

This decision is an amendment to previous decisions related to Concurrent Consent, specifically Decision Proposal 085 and Decision Proposal 099. Community feedback in response to those decision proposals have also been taken into account for this decision.

After the approval of Decision Proposal 085 the Consumer Data Right (CDR) standards support a path for the establishment of multiple active consents between a data recipient, customer and data holder that would minimise implementation impact for July 2020. This decision does not seek to alter that position.

In Decision Proposal 085 a new `existing_refresh_token` claim was added to the request object. In response to consultation feedback it was identified that this was not a preferred solution due to the sharing of a token via the front channel. In addition, the regime has been examining the likely future need for additions to consent such as re-authorisation and fine-grained authorisation.

In response to these needs consultation was conducted under Decision Proposal 099 to determine a solution for November 2020 that would resolve concerns regarding the `existing_refresh_token` claim and lay foundations for the possible future adoption of re-authorisation and fine-grained authorisation.

This solution provides the foundations for a richer consent and authorisation model without pre-supposing a solution before CX research.

Decision To Be Made

Determine a secure and extensible amendment to the solution for concurrent consent which addresses the key concerns and feedback from community consultation. This solution must:

- Be adequately secure
- Allow ADRs to communicate that a new consent is a replacement for an existing consent
- Ensure that the establishment of the new consent and the revocation of the existing consent is atomic
- Provides a technical position that will facilitate future sectors and future use cases

Note that a solution for re-authorisation and for fine grained authorisation is not included in this decision.

Feedback Provided

The feedback leading this decision can be found at:
https://github.com/ConsumerDataStandardsAustralia/open-banking/issues/99

Feedback provided previously on decision proposal 85 was also considered. This can be found at:
https://github.com/ConsumerDataStandardsAustralia/open-banking/issues/85

Based on the consultation there was broad support for:

- Adoption of a Sharing Identifier to represent sharing agreements
- Removal of `existing_refresh_token` claim as a solution
- Adopting Pushed Authorisation Requests (PAR) for making authorisation requests.

During consultation, several concerns were raised where changes were recommended to improve security and useability. These concerns included:

Issue #	Concern / feedback	Mitigation
DP #99	Use of `existing_refresh_token` is not supported	Adoption of a Sharing ID and deprecation of `existing_refresh_token`
Issue #57	Using an authorisation token for managing business concerns	Adoption of a Sharing ID and Sharing Agreement Management API for Data Holder and Data Recipient instead of `existing_refresh_token`
Issue #57, Issue #74	Handling of sensitive information is conducted in the front-channel and should be moved to the backchannel	Adoption of Pushed Authorisation Requests by reference to move communication to the backchannel
Issue #57	Use of oAuth endpoints for managing business concerns	Token revocation endpoint must only be used for token management not consent management
Issue #74	Passing request objects by value has known security and header size issues	Adoption of Pushed Authorisation Requests and JWT Secured Authorization Request (JAR) obviate both issues

In addition to the feedback described above the community provided feedback that was not directly relevant for this consultation but may be useful for consideration in the future when additional consent functionality is considered.

This includes:

- **Use of a Consent API to manage consent:** Currently the CDR regime does not permit the sharing of consent as a resource or the amendment of consent post-authorisation.
- **Rich Authorisation Request (OIDF):** This is an emerging draft to represent rich authorisation permissions. This decision does not seek to address fine-grained authorisation, but this may be considered in future. The decision outlined in this document does not preclude the potential future adoption of this draft specification.
- **Grant Management API (OIDF):** This is an emerging draft to manage grants. Given its draft status it is likely to change. Changes to the draft will be monitored and reviewed from time to time. This may be considered in the future.

- **Client-initiated backchannel authentication (CIBA):** This decision does not require CIBA to meet its requirements. This may be considered in the future.

Decision For Approval

In summary, the standards will be amended as follows:

- Adoption of a `sharing_id` claim returned in the ID Token and accessible via the Data Holder Token endpoint
- Removal of the use of the `existing_refresh_token` claim. This is deprecated and retired in favour of a `sharing_id` claim which would have the same purpose
- Adoption of Pushed Authorisation Requests (PAR) and JWT Secured Authorization Request (JAR)
- A new Sharing Agreement Management API to allow Data Recipients to revoke consent at the Data Holder and vice versa
- Data Holders publishing discovery metadata for PAR and Sharing Agreement Management API support

These changes have a November 2020 Future Dated Obligation.

Sharing Identifier

Introduction of a Sharing Identifier is used to represent an ongoing sharing agreement between a data recipient and data holder for a given consumer. This sharing identifier is represented as a `'sharing_id'` claim that would be issued by Data Holders when a new sharing agreement is established.

For any active consents before concurrent consent obligations, a Data Holder will be required to retrospectively generate a ‘`sharing_id`’. This would mean that all active consents in the CDR ecosystem would have a sharing identifier.

For any active consents before concurrent consent obligations, a Data Recipient will be required to proactively obtain the ‘`sharing_id`’ for all active consents using the token end point.

Implications:

- The sharing identifier is used instead of `existing_refresh_token`
- Use of `existing_refresh_token` is deprecated and must not be supported

Adoption of Pushed Authorisation Requests (PAR)

To facilitate concurrent consent and also be able to move sensitive communications out of the front-channel into the backchannel, PAR must be supported by Data Holders by their concurrent consent obligation dates. This also provides the foundations for a richer consent model in future when fine-grained consent and re-authorisation are in scope.

Data Holders publish their support of PAR as per the PAR normative references by using the OIDC Metadata Discovery endpoint.

Implications:

- Data Holders must support PAR by November 2020 as part of concurrent consent obligations
- The presence of PAR support indicates to Data Recipients that a Data Holder can support concurrent consent
- This support is a substitute for FAPI Pushed Request Object. FAPI Pushed Request Object will not be supported by the CDR standards

Adoption of JWT Secured Authorization Request (JAR) to allow Request Objects by reference

In order to move existing authorisation requests from the front-channel into the backchannel, JAR support allows the Data Recipient to stage an authorisation request and receive a unique `'request_uri'` to complete authorisation.

Data Holders must continue to support request objects sent by value because not all use cases require complex authorisation. A Data Recipient may still send a request object by value in the authorisation flow in situations such as one-time consents where a refresh token is not provisioned and new consent establishment where no existing sharing arrangement exists.

Implications:

- Communication of staged authorisation now occurs via backchannel
- Required dependency for PAR support
- Avoids known header size issues with passing authorisation request objects by value
- Data Holders must support both pushed request objects by value and by reference which introduces their implementation burden

Sharing Agreement Management API

At present, as the refresh token is being used as a proxy to identify the sharing arrangement the data standards only allow for token revocation not sharing arrangement revocation. Effectively this meets the requirements of the rules: A Data Recipient cannot complete a data sharing request after the customer revokes consent. It does, however, represent an overload of the use of the token revocation endpoint.

Introduction of a Sharing Agreement Management API allows Data Recipients and Data Holders to revoke consent via their dashboards along with revoking authorisation tokens.

Moving to a Sharing Agreement Management API allows for more mature notification services related to a sharing agreement between both parties in the future.

Implications:

- Data Recipients must call the Data Holder Sharing Agreement Management API instead of the oAuth Token revocation endpoint to revoke consent
- Data Holders must call the Data Recipient Sharing Agreement Management API where they previously called the Data Recipient Revocation endpoint
- Data Holders and Data Recipients must implement a new API

- Data Recipients must publish a RecipientBaseURI in their Software Statement Assertion
- RecipientBaseURI is a new claim introduced for Data Recipient endpoints

Authorisation Server Metadata & Discoverability

Data Recipients require a way to discover, and in some instances, negotiate with Data Holders. This is handled by the Data Holder making important metadata available via their OpenID Provider discovery endpoint. As per the standards on Pushed Authorisation Requests, Data Holders must publish their PAR endpoint. Similarly, Data Holders will be required to publish their Sharing Agreement Management API endpoint to allow Data Recipients to discover and connect to the endpoint.

Implications:

- Data Holders must publish new claims in their OIDC metadata discovery endpoint
  Data Recipients can infer a Data Holder's support for concurrent consent through the OIDC discovery metadata

Changes to existing standards

Removed Statements

The following statements will be removed from the standards:

Section reference	**Statement**	**Change**
[Request Object]()	*Request Object references SHALL NOT be supported*	Request Object references MUST be supported if the Data Holder supports Pushed Authorisation Requests (PAR).
[Specifying An Existing Refresh Token]()	To allow for an existing consent to be reliably revoked upon the establishment of a new consent intended as a replacement data holders MUST support an additional claim in the authorisation request object named `existing_refresh_token` that the data recipient may optionally include with the value set to the active refresh token for an existing consent. The `existing_refresh_token` claim MUST be handled as follows: Until November 2020 data holders are not required to take any action if `existing_refresh_token` is supplied but MUST NOT respond with an error. From November 2020 data holders MUST revoke a token provided in the `existing_refresh_token` claim in the request object once the new consent is successfully established and a new set of	The `existing_refresh_token` must not be supported. This solution is deprecated in favour of `sharing_id` and the solution components described in this Decision.

	tokens has been provided to the data recipient. Until November 2020 data recipients MUST NOT implement scenarios that support concurrent consent. Only single, extant consent scenarios should be implemented until this date. Until November 2020 data recipients MUST actively revoke previously supplied refresh tokens, immediately after receiving the tokens for a newly established consent, using the revocation end point.	
Revocation End Point	Data Holders and Data Recipients MUST implement a Token Revocation End Point as described in section 2 of **[RFC7009]**.	Data Holders MUST implement a Token Revocation End Point as described in section 2 of **[RFC7009]**.
Revocation End Point	**Requirements for Data Recipient implementations** The Revocation End Point, when implemented by the Data Recipient allows a Data Holder to notify the Data Recipient of the revocation of a sharing arrangement by the Customer in totality as required by the ACCC CDR Rules. This revocation will have been actioned by the Customer via the Data Holder's consent dashboard as described in the ACCC CDR Rules. Revocation of Access Tokens MUST not be supported. Revocation of Refresh Tokens MUST be supported and will be used to notify the Data Recipient of sharing revocation If consent is withdrawn by a Customer in writing or by using the Data Recipient's dashboard the Data Recipient MUST use the Data Holder's implementation of the revocation end point with the current Refresh Token to notify the Data Holder.	Data Recipients must implement the Sharing Agreement Management API

Normative references

PAR - OAuth 2.0 Pushed Authorization Requests (draft-ietf-oauth-par-01)
JAR - The OAuth 2.0 Authorization Framework: JWT Secured Authorization Request (JAR)
RFC8414 - OAuth 2.0 Authorization Server Metadata
IANA.OAuth.Parameters - OAuth Parameters Registry

Sharing Identifier

Statements

- The Sharing ID is a string representing a unique sharing agreement between a data recipient and data holder for a given consumer
- The Sharing ID is represented as a claim "`sharing_id`" in the ID Token
- The Sharing ID **MUST** be unique to a Data Holder
- The Sharing ID **MUST** be non-guessable and must not identify a consumer
- A Sharing ID **MUST** be bound to only one active consent at a time but may have no active consent
- A Sharing ID can span multiple historical consents which are not active
- A Sharing ID **SHOULD** be generated using an algorithm that reduces the chances of collision
- A Sharing ID **MUST** be static across consents within the one sharing agreement (e.g. across consent renewal and re-authorisation)
- A Sharing ID **MUST** be used to revoke consent

Examples:

1. The Issuer creates a Globally Unique Identifier (GUID) [RFC4122] for the pair of Sector Identifier and local sharing ID and stores this value.

2. The local sharing ID and a salt value that is kept secret by the Data Holder. The concatenated string is then hashed using an appropriate algorithm.
   Calculate `sharing_id = SHA−256 (local_sharing_id || salt).`

3. If the Data Holder only provides products to one commercial sector, the Sector Identifier can be concatenated with a local sharing ID and a salt value that is kept secret by the Provider. The concatenated string is then encrypted using an appropriate algorithm.
   Calculate `sharing_id = AES−128 (sector_identifier || local_sharing_id || salt).`

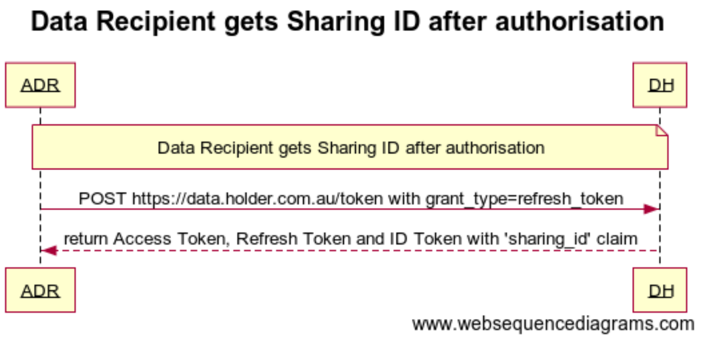
Obtaining a Sharing Identifier

The Data Holder must provide the Sharing ID as a claim in the ID Token as part of a Token endpoint response.

A Data Recipient can call this endpoint at any point post-consent to hydrate an ID Token with the Sharing ID using a valid refresh token.

The sharing ID will be supplied in the ID Token as the claim "`sharing_id`".

Sequence diagram


Data Recipient gets Sharing ID after authorisation
ADR
DH
Data Recipient gets Sharing ID after authorisation
POST https://data.holder.com.au/token with grant_type=refresh_token
return Access Token, Refresh Token and ID Token with 'sharing_id' claim
ADR
DH
www.websequencediagrams.com

Non-normative example

Request

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

client_id=s6BhdRkqt3
&client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-
bearer
&client_assertion=eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjEyNDU2In0.ey ...
&grant_type=refresh_token
&refresh_token=8xLOxBtZp8
&scope=openid%20profile

## Decoded client assertion JWT
{
  "alg": "PS256",
  "typ": "JWT",
  "kid": "12456"
}
{
  "iss": "12345",
  "sub": "12345",
  "iat": 1516239022,
  "exp": 1516239322,
  "aud": "https://data.holder.com.au/token",
  "jti": "37747cd1-c105-4569-9f75-4adf28b73e31"
}
```

Response

```
{
  "iss": "https://data.holder.com.au",
  "sub": "a9ebbef6-1f0b-44eb-96cf-0c5b51b37ab2",
  "aud": "12345",
  "nonce": "n-0S6_WzA2Mj",
  "exp": 1311281970,
  "iat": 1311280970,
  "nbf": 1311280970,
  "auth_time": 1311280969,
  "acr": "urn:cds.au:cdr:3",
  "refresh_token_expires_at": "1311281970",
  "sharing_expires_at": "1311281970",
  "sharing_id": "02e7c9d9-cfe7-4c3e-8f64-e91173c84ecb"
}
```

Supporting Pushed Authorisation Requests by reference

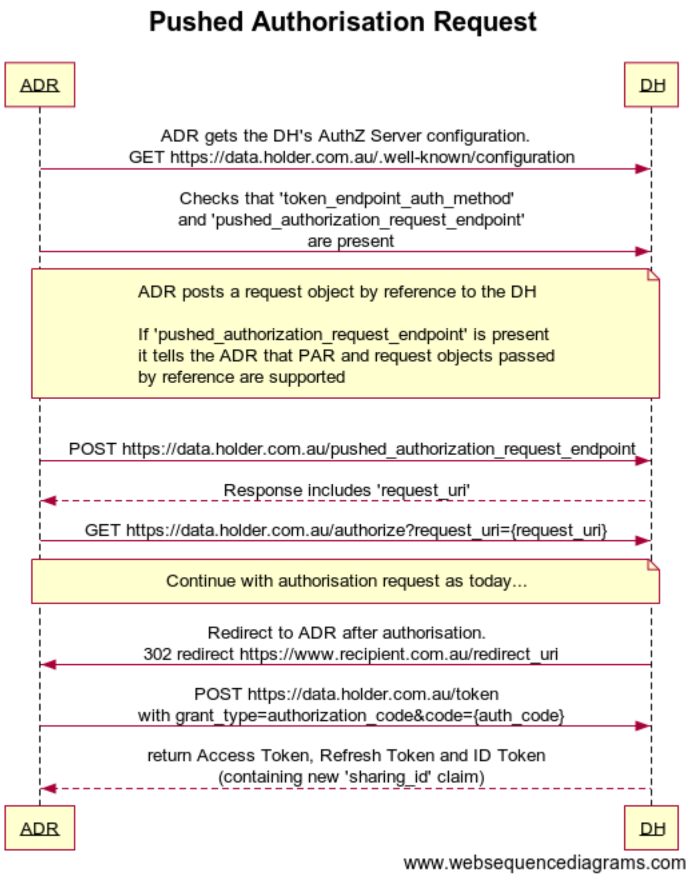
Data Holders must support Pushed Authorisation Requests (PAR) and JWT Secured Authorization Request (JAR).

Data Recipients must send authorisation request objects by reference by calling the Data Holder's pushed authorisation request endpoint if:

- The request object is likely to be too large to be sent as a URI parameter
- The request object contains a `sharing_id` parameter

The Data Holder response provides the Data Recipient with a Request URI in the response. The Request URI is then passed to the Data Holder's Authorisation endpoint to initiate an authorisation flow. In this way, the Data Recipient has staged their authorisation intent with the Data Holder and can then proceed via the backchannel.

Sequence diagram


Pushed Authorisation Request
ADR
DH
ADR gets the DH's AuthZ Server configuration.
GET https://data.holder.com.au/.well-known/configuration
Checks that 'token_endpoint_auth_method' and 'pushed_authorization_request_endpoint' are present
ADR posts a request object by reference to the DH
If 'pushed_authorization_request_endpoint' is present it tells the ADR that PAR and request objects passed by reference are supported
POST https://data.holder.com.au/pushed_authorization_request_endpoint
Response includes 'request_uri'
GET https://data.holder.com.au/authorize?request_uri={request_uri}
Continue with authorisation request as today...
Redirect to ADR after authorisation.
302 redirect https://www.recipient.com.au/redirect_uri
POST https://data.holder.com.au/token
with grant_type=authorization_code&code={auth_code}
return Access Token, Refresh Token and ID Token (containing new 'sharing_id' claim)
ADR
DH
www.websequencediagrams.com

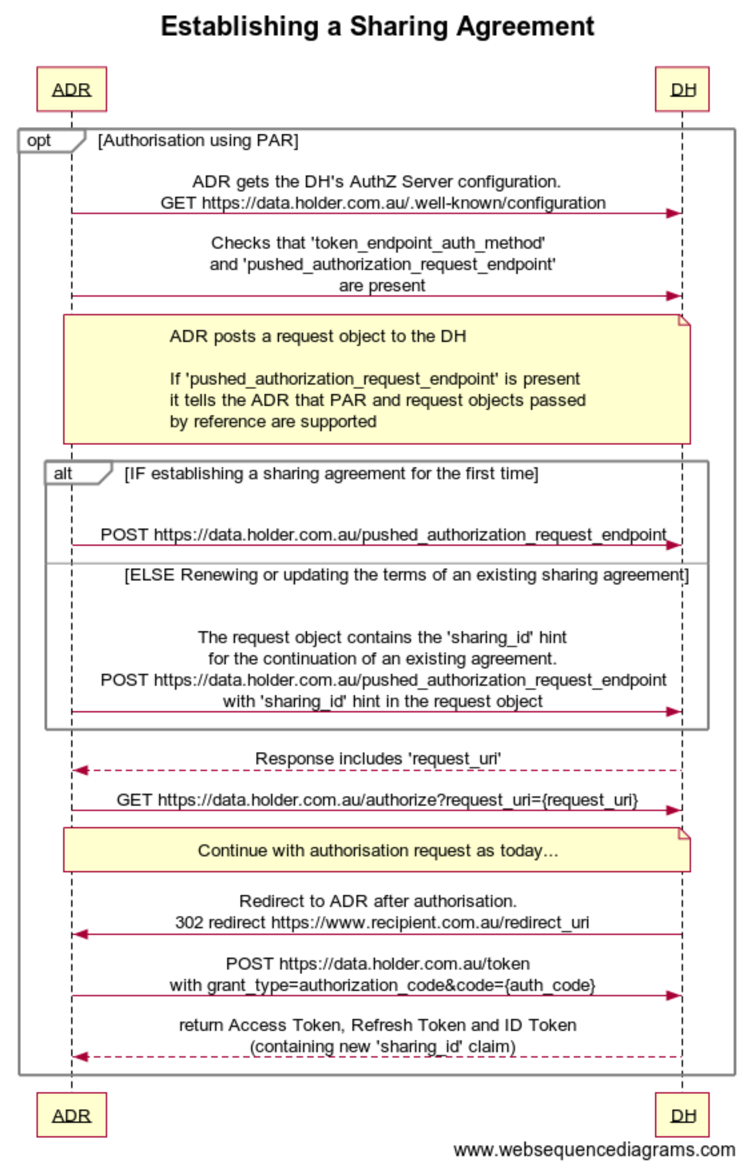
Endpoint

Description	Value
Hosted By	Data Holder
Transport Security	TLS
Client Authentication Required	No
Bearer Token Required	No

Statements

- Data Holders **MUST** support Pushed Authorisation Requests
- Data Holders **MUST** support JAR
- Data Holders **MUST** support request objects sent by reference
- Data Holders **MUST** publish their support for PAR as per the specification using OAuth/OpenID Provider Metadata parameters in discovery responses
- The Request URI **MUST** expire between 10 seconds and 90 seconds
- Data Recipients **MAY** provide an existing `sharing_id` as a hint in an authorisation request object
- Data Holders **MUST** revoke existing authorisation tokens and consents when a `sharing_id` is provided as a hint in the authorisation request object
- Data Recipients **MUST** observe data deletion and de-identification requirements for revoked consent
- If the `sharing_id` is not related to the consumer being authenticated it **MUST** be rejected
- If the `sharing_id` is not related to the Data Holder it **MUST** be rejected

Sequence diagram


Establishing a Sharing Agreement
ADR
DH
opt [Authorisation using PAR]
ADR gets the DH's AuthZ Server configuration.
GET https://data.holder.com.au/.well-known/configuration
Checks that 'token_endpoint_auth_method' and 'pushed_authorization_request_endpoint' are present
ADR posts a request object to the DH
If 'pushed_authorization_request_endpoint' is present it tells the ADR that PAR and request objects passed by reference are supported
alt [IF establishing a sharing agreement for the first time]
POST https://data.holder.com.au/pushed_authorization_request_endpoint
[ELSE Renewing or updating the terms of an existing sharing agreement]
The request object contains the 'sharing_id' hint for the continuation of an existing agreement.
POST https://data.holder.com.au/pushed_authorization_request_endpoint with 'sharing_id' hint in the request object
Response includes 'request_uri'
GET https://data.holder.com.au/authorize?request_uri={request_uri}
Continue with authorisation request as today…
Redirect to ADR after authorisation.
302 redirect https://www.recipient.com.au/redirect_uri
POST https://data.holder.com.au/token
with grant_type=authorization_code&code={auth_code}
return Access Token, Refresh Token and ID Token
(containing new 'sharing_id' claim)
ADR
DH
www.websequencediagrams.com

Non-normative example

Request

```
## Request

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

response_type=code%20id_token
 &client_id=12345
 &redirect_uri=https%3A%2F%2Fwww.recipient.com.au%2Fcoolstuff
 &scope=openid%20profile%20bank:accounts.basic:read
        %20bank:accounts.detail:read
 &nonce=n-0S6_WzA2Mj
 &state=af0ifjsldkj
 &request=eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjEyMyJ9.ey...
```

Response

```
## Response

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

{
  "request_uri": "urn:data.holder.com.au:bwc4JK-ESC0w8acc191e-Y1LTC2",
  "expires_in": 3600
}
```

Sharing Agreement Management API and consent revocation

If a Data Recipient wishes to revoke consent, it must do so by calling the Data Holder's sharing agreement revocation endpoint.

Data Recipients must use a valid Access Tokens as specified in `section 10.3` of [OAUTH2]

Endpoint

VERBs	DELETE

API	https://data.holder.com.au/sharing/{sharing_id}
	https://data.recipient.com.au/sharing/{sharing_id}

Description	Value
Hosted By	Data Holder and Data Recipient
Transport Security	MTLS
Client Authentication Required	No
Bearer Token Required	Yes

Race conditions and handling consent revocation with Data Recipients

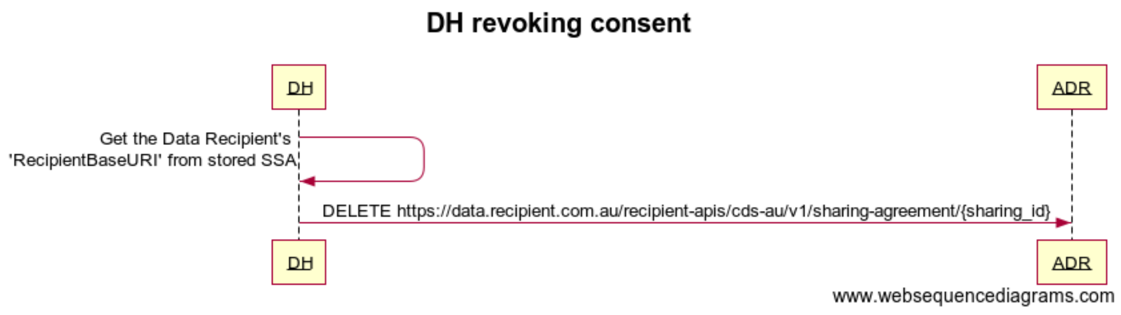
Because single-consent sharing agreements will be established before concurrent consent future dated obligations, there is the chance that a consumer may revoke consent with a Data Holder before a Data Recipient has obtained a Sharing ID. In this instance, a Data Holder will call the Data Recipient’s Sharing Agreement Management API with a Sharing ID that is not recognised by the Data Recipient. The Data Recipient would return an error which signifies to the Data Holder that the `sharing_id` is not recognised.

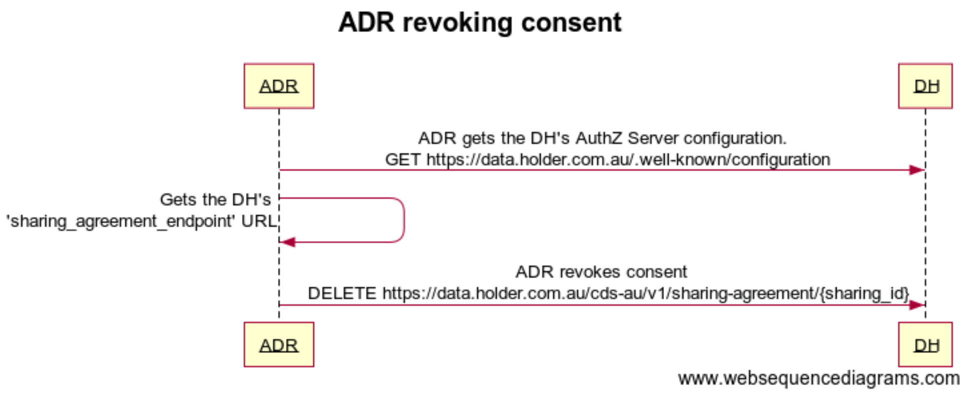
In this instance, a Data Holder must attempt to call the Data Recipient’s revocation endpoint to notify the Data Recipient that a sharing agreement has ended. If the Data Recipient has chosen to no longer support a revocation endpoint, the absence of support will be inferred through the absence of the `revocation_endpoint` in the Data Recipients software statement assertion (SSA).

Statements

- Consent management **MUST** be managed though the new Sharing Agreement Management API. The Sharing Agreement Management API only supports DELETE for revocation of consent for the scope of concurrent consent.
- Data Recipients and Data Holders **MUST** revoke consent by calling the Sharing Agreement Management API with a valid sharing identifier
- Data Holders **MUST** publish their Sharing Agreement Management API using their OpenID Provider Metadata discovery endpoint
- Data Recipients **MUST** publish their Sharing Agreement Management API under their InfoSec Base URI published in the CDR Register
- If the Sharing Agreement Management API is called for revocation, it **MUST** delete associated authorisation tokens
- Data Recipients **MAY** deprecate support of the revocation endpoint. This **MUST** be inferred through the absence of the `revocation_endpoint` in the Data Recipients SSA.
- The Data Recipient's Revocation endpoint **MUST ONLY** revoke authorisation tokens

Sequence diagrams


DH revoking consent
DH
ADR
Get the Data Recipient's 'RecipientBaseURI' from stored SSA
DELETE https://data.recipient.com.au/recipient-apis/cds-au/v1/sharing-agreement/{sharing_id}
DH
ADR
www.websequencediagrams.com


ADR revoking consent
ADR
DH
ADR gets the DH's AuthZ Server configuration.
GET https://data.holder.com.au/.well-known/configuration
Gets the DH's 'sharing_agreement_endpoint' URL
ADR revokes consent
DELETE https://data.holder.com.au/cds-au/v1/sharing-agreement/{sharing_id}
ADR
DH
www.websequencediagrams.com

Non-normative example
Request

```
DELETE https://data.holder.com.au/sharing/5a1bf696-ee03-408b-b315-97955415d1f0
HTTP/1.1
Host: data.holder.com.au
Authorization: Bearer
eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjEyNDU2In0.ey...
x-v: string
x-min-v: string
x-fapi-interaction-id: string
x-fapi-auth-date: string
x-fapi-customer-ip-address: string
x-cds-client-headers: string
```

Response
The Data Holder responds with HTTP status code 204 if the sharing agreement has been revoked successfully or if the client submitted an invalid token.

Refresh Token management

Currently, consent revocation is handled by calling the Data Holder's oAuth token revocation endpoint. From November 2020, this will only be allowed by using an `existing_refresh_token` and the overloaded use of the Data Holder's oAuth token revocation endpoint. Because the token revocation endpoint should only be used for oAuth token management, revocation of consent cannot rely on token revocation because this couples business and security concerns. As a result, a solution that decouples these concerns is necessary. The Sharing ID in conjunction with a Sharing Agreement Management API supports the decoupling of these concerns such that consent revocation can be performed independent of token management.

Effect of token expiry on a sharing agreement's state
A Data Holder may issue an access token and refresh token for a long-lived consent. These tokens may expire before the consent expires. In such a situation, the state of the consent's *intent* does not change, and the Data Holder **must not** modify the state of the intent.

Practically, an ADR presenting a stale access token and/or refresh token would be denied by the Data Holder because their access to the protected resource(s) is no longer current.
It is recommended that a Data Holder records a separate authorisation status for a consent that represents the state of token validity in relation to the consent. The consent status would only change if:

- It has been explicitly revoked (by a consumer either in writing, via the ADR dashboard or via the DH dashboard)
- It has expired after the data `sharing_duration`
- The ADR's status in the register requires consents to be revoked

Statements

- Use of `existing_refresh_token` is deprecated and **MUST NOT** be implemented by Data Holder's as part of November 2020 obligations
- oAuth Token Revocation endpoints **MUST** only be used for the purposes of token management

Discovery Metadata

Data Recipients need a way to discover, and in some instances, negotiate with Data Holders. This is handled by the Data Holder making important metadata available via their OpenID Provider discovery endpoint.

Data Holder Statements

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]**.

Data Holders **MUST** include the following parameters along with any requirements as part of underlying specifications:

- `sharing_agreement_endpoint`: the location of the Data Holder's sharing API for consent revocation
  `pushed_authorization_request_endpoint`: the location of the Data Holder's PAR endpoint per Pushed Authorisation Request

Non-normative example

```
## Data Recipient Request
GET /.well-known/openid-configuration HTTP/1.1
Host: data.holder.com.au

## Data Holder Response
HTTP/1.1 200 OK
Content-Type: application/json
{
  "issuer": "https://data.holder.com.au",
  "authorization_endpoint": "https://data.holder.com.au/authorise",

   ...

  ## Pushed Authorisation Request metadata – mandatory if concurrent
     consent is supported
  "pushed_authorization_request_endpoint":
                                       "https://data.holder.com.au/par",

  ## Location of the sharing API for consent management
  "sharing_agreement_endpoint":
                          "https://data.holder.com.au/sharing-agreement/"
}
```

Data Recipient Statements

- Data Recipients **MUST** publish their Sharing Agreement Management API under the ResourceBaseURI that is published on the CDR Register.

Non-normative example

```
## Data Recipient Sharing Agreement Management API
https://<ResourceBaseUri>/sharing-agreement

## Some example URIs that meet this standard are:
https://data.recipient.com.au/sharing-agreement
https://www.energycompare.com.au/cds-au/v1/api/sharing-agreement
```