Data Standards Body

Technical Working Group

Decision 99 – Finalisation of concurrent consent

Contact: Mark Verstege

Publish Date: 14th April 2020

Decision Approved By Chairman: 17th April 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

Feedback has been provided over the course of three consultations:

1. **22nd September 2019**: Decision Proposal 085 https://github.com/ConsumerDataStandardsAustralia/standards/issues/85
2. **4th February 2020**: Decision Proposal 099 request for feedback https://github.com/ConsumerDataStandardsAustralia/standards/issues/99#issue-559369359
3. **26th March 2020**: Decision Proposal 099 solution proposal document and 2 week consultation https://github.com/ConsumerDataStandardsAustralia/standards/files/4384751/Decision.Proposal.99.-.Concurrent.Consent.pdf

Feedback for consultations the first two consultations have previously been summarised in Decision Proposal 099. This decision document summarises feedback provided during the last two-week consultation period.

Changes supported:

1. Broad recognition and support of CDR's intent to move towards FAPI 2.0 target state
2. Strong support for removing the use of the existing_refresh_token and removal of sensitive communications in the front channel
3. There is broad support for an identifier to represent consent but disagreement on the use of a CDR-specific "CDR Arrangement Identifier" because it does not support future write-operation extensibility
4. Leveraging the PAR industry standard is supported but does not currently have vendor availability and banks do not support implementation within November 2020 timeframes
5. A Consent API in some form is generally supported for querying the status of consent and revoking consent
6. There is strong support for adoption of PKCE in future phases to replace OIDC Hybrid Flow, noting that this aligns to a future roadmap for FAPI 2.0 supportability

Changes unsupported:

1. Vendors and banks provided feedback that the November 2020 dates would be difficult to meet in the current climate because of COVID19
2. There is limited support of a Sharing Agreement API for Data Recipients because of the complexity to Data Recipients
3. Whilst participants support an identifier to represent consent there is disagreement on the use of a CDR-specific "CDR Arrangement Identifier" because it does not support future write-operation extensibility.
   Specifically feedback was received that the CDR Arrangement ID is limiting from the perspective that it only supports read operations within the CDR without being suitably flexible for future use cases (E.g. write operations) that have been earmarked as part of the second Farrell Review into the Future Directions of the CDR.
4. Some banks prefer not to support request objects by value for consistency and simplicity
5. There is limited support of the JAR industry standard and advice is to remove this requirement until it is known to be required in the future

Additional feedback provided

1. Replace use of cdr_arrangement_id and adopt Grant Management standard's grant_id
2. Adopt MTLS for the PAR endpoint
3. Feedback has also been provided where it makes the new statements to support concurrent consent clearer for implementation
4. Adoption of a simplified RAR approach to manage CDR-specific claims in a standards-aligned structure

Executive summary of changes adopted from feedback to Decision Proposal 099

The changes adopted based on feedback from the community do not materially change the conceptual architecture and broad support for the foundations of concurrent consent. The changes provide solution equivalence with closer alignment to industry standards that will reduce implementation costs for the CDR in the long term.

1. **Sharing Agreement Management API**
   - Data Holders and Data Recipients adopt basic consent management API DELETE operation only for consent revocation
   - Sharing Agreement Management API is renamed CDR Arrangement Management API to allow for use beyond read-only data sharing

2. **Use CDR Arrangement ID not Sharing Identifier**
   - Adopt "cdr_arrangement_id" to support use cases beyond data sharing arrangements such as write-operations in future.

3. **Pushed Authorisation Requests**
   - Continue to support requirement of PAR
   - Require MTLS for PAR endpoint
   - Remove adoption of JAR for concurrent consent other than the introduction of 'request_uri' functionality that allows clients to send a reference to a request object instead of the request object itself

4. **Backwards compatibility for existing consents**
   - Remove sharing_id from the ID Token
   - Include cdr_arrangement_id in Token and Token Introspection JWTs instead of sharing_id in ID Token

Consideration of feedback not adopted

It is acknowledged that good feedback was provided on the phasing in of RAR and the Grant Management API, both drafts being developed by the OpenID Foundation. The DSB will continue to review these with the intent to move towards these standards in the future.

Whilst they are not supported within the concurrent consent timeframes and scope, consultation on their phased introduction as part of a broader FAPI 2.0 alignment and CDR Consent roadmap will commence in future.

It was felt that adopting RAR as an aspect of the concurrent consent solution did not have sufficient prior consultation and consideration of all implications to production rollout.

Similarly, with the Grant Management API, it was determined that both Grant Management and CDR's consent model requirements are still emerging and further testing of both solutions together is required.

Decision For Approval

The standards will be amended to change the mechanisms that allow concurrent consents to exist.

In addition, the data recipient will be provided with the ability to specify the CDR arrangement identifier for an existing consent in the authorisation request object in the same way that sharing duration is currently specified. This will provide a mechanism for data recipients to differentiate between a new, concurrent consent, and an amended consent that is intended to supersede a previously established consent.

To be clear, if a cdr_arrangement_id *is not* provided then a new, concurrent consent is established in addition to any existing consents. Existing consents are unaffected.

If a cdr_arrangement_id *is* provided then, upon successful authorisation, the data holder would revoke the existing consent associated with the provided refresh token. In addition, the expiration of sharing would be calculated as the addition of the specified sharing duration to the expiration time of the current consent rather than to the time of authorisation. This would allow for an existing consent to be extended for the full twelve-month allowable period.

Future Dated Obligations

Decision 085 – Concurrent Consent introduced a Future Dated Obligation of November 2020. This obligation date has been retained in alignment with other obligation dates and the existing advertised implementation schedule for the CDR regime. It is acknowledged, however, that the ACCC is currently reviewing the CDR implementation schedule in light of the recent COVID19 pandemic. The Future Dated Obligations for the data standards will be revised to align with the ACCC's determinations in this regard when they are made known.

CDR Arrangement ID (previously Sharing Identifier)

The standards introduce a new CDR Arrangement ID (previously referred to as a Sharing Identifier) in the form of a `'cdr_arrangement_id'` claim.

Introduction of a CDR Arrangement ID is used to represent an ongoing sharing arrangement between a data recipient and data holder for a given consumer. The CDR Arrangement ID would be issued by Data Holders when a new sharing arrangement is established.

For any active consents before concurrent consent obligations, a Data Holder will be required to retrospectively generate a `'cdr_arrangement_id'`. This would mean that all active consents in the CDR ecosystem will have a Grant ID.

For any active consents before concurrent consent obligations, a Data Recipient will be required to proactively obtain the `'cdr_arrangement_id'` for all active consents using either the token or token introspection end point.

Implications:

- The CDR Arrangement ID is used instead of `existing_refresh_token`
- Use of `existing_refresh_token` is deprecated and must not be supported
- For concurrent consent, ONLY the Data Holder may generate a cdr_arrangement_id

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 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

Based on community feedback, inclusion of some aspects of JAR are not required and would create additional implementation complexity. It is noted that the aspect of JAR that are considered important and necessary is the introduction of the "request_uri" parameter that allows clients to send a reference to a request object instead of the request object itself.

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

CDR Arrangement 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 CDR Arrangement Management API allows Data Recipients and Data Holders to revoke consent via their dashboards along with revoking authorisation tokens.

Moving to a CDR Arrangement Management API allows for more mature notification services related to a sharing arrangement between both parties in the future.

Implications:

- Data Recipients must call the Data Holder CDR Arrangement Management API instead of the oAuth Token revocation endpoint to revoke consent
- Data Holders must call the Data Recipient CDR Arrangement 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 CDR Arrangement 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 `existing_refresh_token` must not be supported. This solution is deprecated in favour of `cdr_arrangement_id`

	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 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.	and the solution components described in this Decision.
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	Data Recipients must implement the CDR Arrangement Management API

	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.	

Normative references

PAR - OAuth 2.0 Pushed Authorization Requests (draft-ietf-oauth-par-01)
JAR - JWT Secured Authorization Request (draft-ietf-oauth-jwsreq-20)
RFC8414 - OAuth 2.0 Authorization Server Metadata
IANA.OAuth.Parameters - OAuth Parameters Registry

CDR Arrangement ID

Statements

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

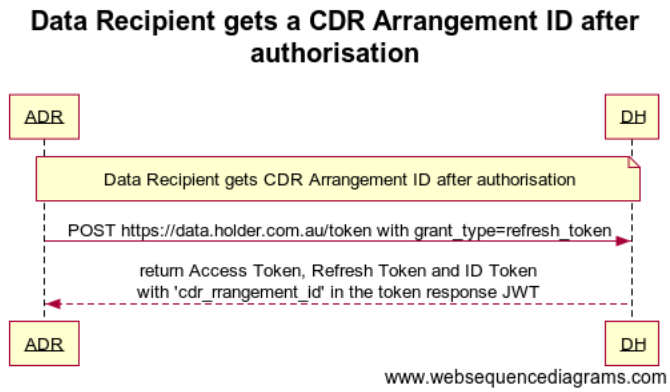
Retrospectively obtaining a CDR Arrangement ID

The Data Holder must provide the CDR Arrangement ID as a claim in the Token endpoint response and Token Introspection endpoint response.

A Data Recipient can call either the Token or Token Introspection endpoints at any point post-consent to obtain the CDR Arrangement ID using a valid refresh token.

The CDR Arrangement ID will be supplied in the response JSON as the claim "`cdr_arrangement_id`".

Sequence diagram



Non-normative example: Token Endpoint hydration

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

```
{
  "access_token": "2YotnFZFEjr1zCsicMWpAA",
  "expires_in": 3600,
  "refresh_token": "tGzv3JOkF0XG5Qx2TlKWIA",
  "id_token": "eyJraWQiOiIxZTlnZGs3IiwiYWxnIjoiUl...",

## DECISION PROPOSAL 099
  "cdr_arrangement_id": "02e7c9d9-cfe7-4c3e-8f64-e91173c84ecb"
}

## Decoded JWT
```

```
{
  "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",

## DECISION PROPOSAL 099
  "cdr_arrangement_id": "02e7c9d9-cfe7-4c3e-8f64-e91173c84ecb"
}
```

Non-normative example: Token Introspection Endpoint hydration

Request

```
POST /token/introspect 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

## 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/introspect",
  "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",

## DECISION PROPOSAL 099
  "cdr_arrangement_id": "02e7c9d9-cfe7-4c3e-8f64-e91173c84ecb"
}
```

Supporting Pushed Authorisation Requests by reference

Data Holders must support Pushed Authorisation Requests (PAR).
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 `cdr_arrangement_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

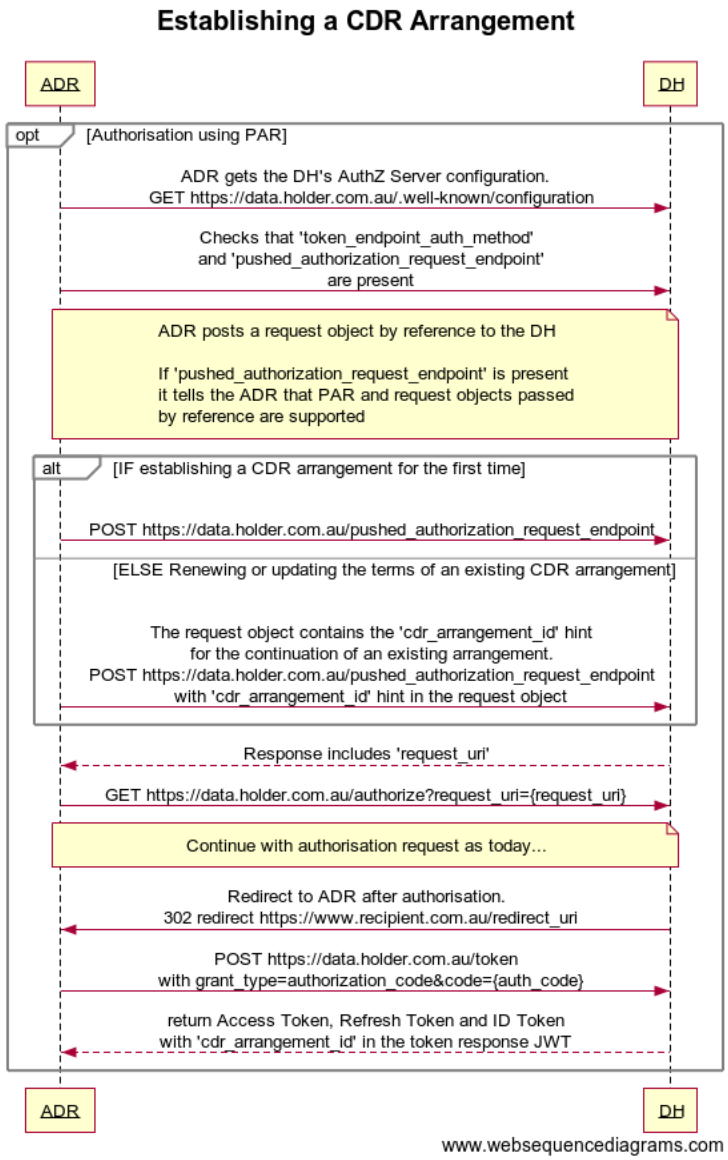
Endpoint

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

Statements

- Data Holders **MUST** support Pushed Authorisation Requests
- Data Holders **MUST** support request objects sent by reference for Pushed Authorisation Requests
- Request Object references **SHALL NOT** be supported in any mode of use other than Pushed Authorisation Requests (PAR). If a Data Holder does not support Pushed Authorisation Requests (PAR), it **MUST NOT** support Request Object references.
- 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 `cdr_arrangement_id` claim in an authorisation request object
  to establish a new consent under an existing arrangement
- Data Holders **MUST** revoke existing refresh tokens and access tokens when a `cdr_arrangement_id` is provided in the authorisation request object but ONLY after successful authorisation
- Data Recipients **MUST** observe data deletion and de-identification requirements for revoked consent after successful authorisation
- If the cdr_arrangement_id is not related to the consumer being authenticated it **MUST** be rejected
- If the `cdr_arrangement_id` is not related to the Data Holder it **MUST** be rejected

Sequence diagram



Non-normative example

Request

```
## Request

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

request=eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjEyMyJ9.ey...

## Decoded Request
{
  "iss": "https://www.holder.com.au",
  "aud": "a7AfcPcsl2",
  "exp": 1311281970,
  “client_id”: s6BhdRkqt3,
  “grant_management_mode”: create,
  “response_type”: ”code id_token”,
```

```
  “client_id”: 12345,
  “redirect_uri”: ”https://www.recipient.com.au%2Fcoolstuff”,
  “scope”: “openid profile bank:accounts.basic:read
            bank:accounts.detail:read”,
  “nonce”: “n-0S6_WzA2Mj”,
  “state”: “af0ifjsldkj”
}
```

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
}
```

CDR Arrangement Management API and consent revocation

If a Data Recipient wishes to revoke consent, it must do so by calling the Data Holder's sharing arrangement 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/arrangements/{cdr_arrangement_id} https://data.recipient.com.au/arrangements/{cdr_arrangement_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 arrangements 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 CDR Arrangement Management API with a CDR Arrangement ID that is not recognised by

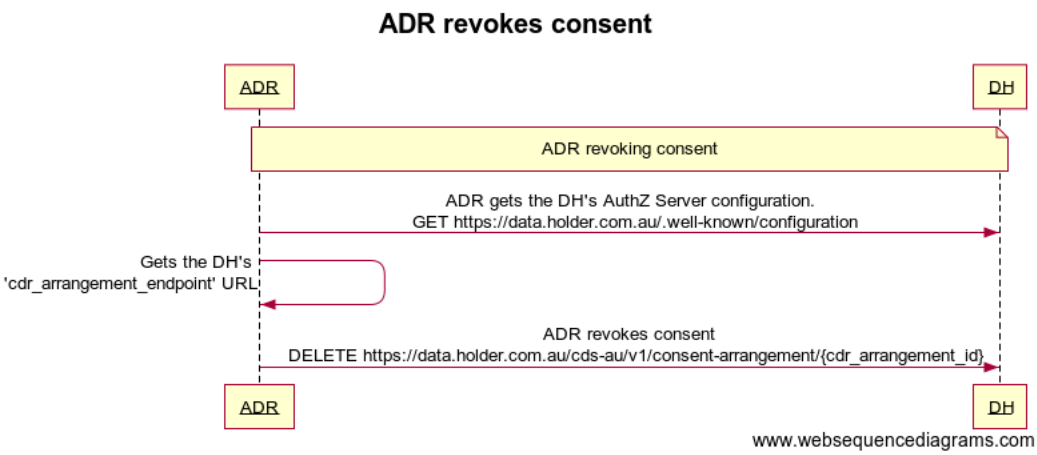
the Data Recipient. The Data Recipient would return an error which signifies to the Data Holder that the `cdr_arrangement_id` is not recognised.

In this instance, a Data Holder must attempt to call the Data Recipient's revocation endpoint to notify the Data Recipient that a sharing arrangement 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 through the new CDR Arrangement Management API. The CDR Arrangement 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 CDR Arrangement Management API with a valid CDR Arrangement Identifier
- Data Holders **MUST** publish their CDR Arrangement Management API using their OpenID Provider Metadata discovery endpoint
- Data Recipients **MUST** publish their CDR Arrangement Management API under their Recipient Base URI published in their Software Statement Assertion
- If the CDR Arrangement Management API is called for revocation, it **MUST** delete associated refresh and/or access tokens
- The Data Recipient's Revocation endpoint **MUST ONLY** be used for the purposes of revoke refresh tokens and/or access tokens
- If the `cdr_arrangement_id` is not related to the consumer being authenticated it **MUST** be rejected
- If the `cdr_arrangement_id` is not related to the Data Holder it **MUST** be rejected

Sequence diagrams


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

Non-normative example

Request

```
DELETE https://data.holder.com.au/consent-arrangement/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 arrangement 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 CDR Arrangement ID in conjunction with a CDR Arrangement ManagementAPI supports the decoupling of these concerns such that consent revocation can be performed independent of token management.

Effect of token expiry on a sharing arrangement'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:

- `cdr_arrangement_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
  "cdr_arrangement_endpoint":
                         "https://data.holder.com.au/consent-arrangement/"
}
```

Data Recipient Statements

- Data Recipients **MUST** publish their CDR Arrangement ManagementAPI under the ResourceBaseURI that is published on the CDR Register.

Non-normative example

```
## Data Recipient CDR Arrangement ManagementAPI
https://<ResourceBaseUri>/consent-arrangement

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