|  |
| --- |
|  |
| Consumer Data Right Conformance Test Suite |
| Data Holder (DH)Technical Guidance Material  |
| February 2022 CTS V3.4.1 |
|  |

# Document Control

|  |  |
| --- | --- |
| Document Version | CTS v3.4.1 |
| Document Status | Published |
| Issued Date | 02/03/2022 |
| Owner | ACCC |

Revision History

|  |  |  |
| --- | --- | --- |
| Version | Date | Description of Changes |
| CTS v2.0 | 1/3/2021 | Initial Version  |
| CTS v2.0.1 | 25/3/2021 | * Rename “Inactive Software Product” to “Reactivate software product”
* Updated Scenario 7 for error on step 9b
* Updated Scenario 7 to include the missing steps to Reactivate the Software Product
* Update all Scenario Diagrams
 |
| CTS v3.0.0 | 15/4/2021 | Added 2 new scenarios:* Ensure Client Assertion Data in Token Request
* Amending Account for An Existing Consent Scenario with PAR
* Maintenance updates to accommodate elements from CDS 1.7.0 and Register design 1.3.0
 |
| CTS v3.1.0 | 29/4/2021 | Added 3 new scenarios:* Ensure Holder of Key for Resource Requests
* Ensure Infosec Endpoints using MTLS authentication with X509 Certificates
* Consent Software Statement Assertion with sector identifier uri
* Removed URI information from individual scenarios and centralised in a table in a later section
* Added a phasing table in a later section
 |
| CTS v3.1.1CTS v3.2CTS v3.3 | 18/5/202121/6/202123/07/2021 | Updated document version number and title page to align with CTS v3.1.1Updates to the document aligning the version to sync with ADR and the formatting.Changes to the Concurrent Consent scenario to reflect phase 2 authorisation scopes and end points. |
| CTS v3.3 | 16/08/2021 | Minor change to scenario numbering structure to ensure polling variants match system test plans numbering |
| CTS v3.4 | 02/09/2021 | Updates to the following scenarios to cater for CTS v3.4 and standard error handling changes as per CDS v1.10.0+:* Scenario 6 - Removed Software Product
* Scenario 7 - Reactivate Software Product
 |
| CTS v3.4 | 20/09/2021 | Minor updates to commentary on Scenario 6 (Step 7) and Scenario 7 (Step 8).  |
| CTS v3.4.1 | 20/01/2022 | Updates on CTS Register endpoints to be consistent with CDR Register endpoints, no operational process change. Details of CTS Register endpoint changes are:**Register Discovery Document**:https://api.{env}.cts.cdr.gov.au/cts/{conformanceId}/register/idp/.well-known/openid-configuration**Register Token Endpoint**:https://secure.api.{env}.cts.cdr.gov.au/cts/{conformanceId}/register/idp/connect/token**Register Token Keyset**: https://api.{env}.cts.cdr.gov.au/cts/{conformanceId}/register/idp/.well-known/openid-configuration/jwks**Register SSA Keyset**https://api.{env}.cts.cdr.gov.au/cts/{conformanceId}/register/cdr-register/v1/jwks |
|  |  |  |

Table of Contents

[Consumer Data Right Conformance Test Suite 0](#_Toc78537923)

[Document Control 1](#_Toc78537924)

[1. Overview 5](#_Toc78537925)

[Document Purpose 5](#_Toc78537926)

[Background 5](#_Toc78537927)

[CTS Scope for DH 5](#_Toc78537928)

[2. DH CTS Scenarios and Tests 7](#_Toc78537929)

[Scenario 1 – Discovery Document Validation 9](#_Toc78537930)

[Scenario High Level Test Steps 9](#_Toc78537931)

[Scenario 2 - Dynamic Client Registration (DCR) 11](#_Toc78537932)

[Scenario High Level Test Steps 12](#_Toc78537933)

[Scenario 3 – Concurrent Consent 14](#_Toc78537934)

[Scenario High Level Test Steps 15](#_Toc78537935)

[Scenario 4 – DH Initiated Revocation 21](#_Toc78537936)

[Scenario High Level Test Steps 21](#_Toc78537937)

[Scenario 5 – DR Initiated Revocation 23](#_Toc78537938)

[Scenario High Level Test Steps 23](#_Toc78537939)

[Scenario 6 – Removed Software Product 25](#_Toc78537940)

[Scenario High Level Test Steps 26](#_Toc78537941)

[Scenario 7 – Reactivate Software Product 28](#_Toc78537942)

[Scenario High Level Test Steps 29](#_Toc78537943)

[Scenario 8 – Replace Existing Consent with PAR 32](#_Toc78537944)

[Scenario High Level Test Steps 33](#_Toc78537945)

[Scenario 9 – Register PUT GET 35](#_Toc78537946)

[Scenario High Level Test Steps 35](#_Toc78537947)

[Scenario 10 – Token Revocation 37](#_Toc78537948)

[Scenario High Level Test Steps 37](#_Toc78537949)

[Scenario 11 – Get Software Product Status Register Polling 40](#_Toc78537950)

[Scenario High Level Test Steps 40](#_Toc78537951)

[Scenario 12 – Get Data Recipients Register Polling 42](#_Toc78537952)

[Scenario High Level Test Steps 42](#_Toc78537953)

[Scenario 13 – Ensure Client Assertion Data in Token Request 44](#_Toc78537954)

[Scenario High Level Test Steps 44](#_Toc78537955)

[Scenario 14 – Amending Account for an Existing Consent Scenario with PAR 49](#_Toc78537956)

[Scenario High Level Test Steps 50](#_Toc78537957)

[Scenario 15 – Ensure Holder of Key (HoK) for Resource Requests 52](#_Toc78537958)

[Scenario High Level Test Steps 52](#_Toc78537959)

[Scenario 16 – Ensure Infosec Endpoints Using MTLS Authentication with X509 Certificates 54](#_Toc78537960)

[Scenario High Level Test Steps 54](#_Toc78537961)

[Scenario 17 – Consent Software Statement Assertion with Sector Identifier URI Scenario 56](#_Toc78537962)

[Scenario High Level Test Steps 57](#_Toc78537963)

[3. API’S 60](#_Toc78537964)

[4. Endpoints used in DH Scenarios 61](#_Toc78537965)

[5. Glossary 63](#_Toc78537966)

[6. Brand vs Conformance ID Infographic 66](#_Toc78537967)

# Overview

## Document Purpose

The purpose of this document is to provide technical information about the Consumer Data Right (CDR) Conformance Test Suite (CTS) (the CTS). It will provide an in depth understanding for Data Holder Brands and Data Recipient software products on:

* the scope of the CTS
* the purpose of each CTS scenario/test
* what is being tested to provide conformance
* how to make and respond to each CTS scenario/test
* how to react correctly to valid and invalid requests.

## Background

The CTS is an important part of the Consumer Data Right on-boarding process. It is designed to check conformance against a set of CDR scenarios/tests. Participants must pass the CTS before their brand or software product becomes active on the Register.

The CTS is not a sandbox or assisted development tool. It will not help you design and build a product that conforms to the CDS and the Register design.  Before beginning conformance testing, you should be confident your brand or software product will pass the CTS.

For the steps you need to complete before you can use the CTS please consult the CTS Guidance Material.

## CTS Scope for DH

The CTS interacts with the DH Brand and assess the Data Holder’s technical competency in conforming to the CDS. To achieve this, the CTS simulates the Register and a Data Recipient, testing that the DH Brand can safely interact with a data recipient in the system.

|  |  |
| --- | --- |
| In Scope | Out of Scope |
| The CTS will conduct a series of tests to determine the technical competency of the DH as to whether they can conform to the CDS and the Register design.  * Dynamic client registration
* Interaction with the Register
* Consent
* Banking APIs (Accounts)
* Register Status
* Consent Withdrawal
 | The CTS does not test the internal workings and validations of DH Brands.* how consent is managed within a DHs Brand
* how a DHs Brand correctly handles certain consent flow attack vectors
* how a DH removes consent and consumer data in their Brand
 |
| Note: The CTS includes only those endpoints that are detailed in this document. (Note: there is not a 1:1 relationship with RAAP) |

# DH CTS Scenarios and Tests

This section captures the following scenarios and all their associated steps that can be part of a DH Test Plan:

1. Discovery Document Validation
2. Dynamic Client Registration (DCR)
3. Concurrent Consent
4. DH Initiated Revocation
5. DR Initiated Revocation
6. Removed Software Product
7. Reactivate Software Product
8. Replace Existing Consent with PAR Scenario
9. Register PUT GET
10. DR Initiate Token Revocation
11. Get Software Product Status Register Polling [[1]](#footnote-2)
12. Get Data Recipients Register Polling[[2]](#footnote-3)
13. Ensure Client Assertion Data In Token Request
14. Amending Account for An Existing Consent Scenario with PAR
15. Ensure Holder of Key (HoK) for Resource Requests
16. Ensure Infosec Endpoints Using MTLS Authentication with X509 Certificates
17. Consent Software Statement Assertion with Sector Identifier URI Scenario

**CTS Entry Criteria**

The CTS simulates both the Register and an ADR that your brand interacts with. You should enrol in the CTS when your Brand is ready for production release or close to being ready.  After receiving your enrolment confirmation, you can start your CTS tests. You are advised to take the CTS tests in the order they are listed in this document.

**Before you start:**

1. apply the test certificate to your Brand
2. you must have a valid account on the CDR Participant Portal
3. you must be registered as a CTS tester as part of the CTS enrolment.

**CTS Exit Criteria**

1. You must execute the tests as selected in your enrolment form.
2. This test can be run multiple times during the test run. The result of the last attempt of the test will be included in the test run report for CTS outcome assessment.
3. You are required to provide test results for all scenarios on your test plan, or to provide justification on why the test/s is not relevant. ACCC can give special consideration on whether to grant a CTS Pass status even if you fail a test.
4. Submit the test result via the DH UI after finishing the tests. You must inform the On-boarding Officers when you submit your test results via email so that the On-boarding Officer can start assessing their results.

## Scenario 1 – Discovery Document Validation

**Purpose**

The ability to validate a DHs Discovery Document to support several test scenarios which use information provided in the Discovery Document.

**Scenario Conditions**

* This is a mandatory test.
* Discovery Document steps are part of most of the scenarios.

**Endpoints**

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Description | Method | Uri |
| Discovery (DH) | CTS ADR requests the discovery document from the Data Holder via the Discovery Endpoint | GET | [Endpoint URI](#_Endpoints_used_in) |

**Link to specs**

<https://consumerdatastandardsaustralia.github.io/standards/#end-points>

<https://consumerdatastandardsaustralia.github.io/standards/#openid-provider-configuration-end-point>

<https://openid.net/specs/openid-connect-discovery-1_0.html>

**Scenario Results**

**Pass**: you have achieved a successful CTS Discovery Document test when CTS sends a get request to the DH Discovery Endpoint (Provided during enrolment) AND retrieves and validates the Discovery Document JSON response.

**Fail**: you have failed the CTS Discovery Document test when CTS sends a get request to the DH Discovery Endpoint (Provided during enrolment) AND is not able to retrieve or successfully validate the Discovery Document JSON response.

### Scenario High Level Test Steps

#### Discovery Document Validation

1. CTS ADR requests a Discovery Document from the DH
2. CTS ADR sends a Discovery Document request to the DH via the OpenID Provider Configuration end point.
3. DH returns a response with their Discovery Document.
4. CTS verifies the Discovery Document



## Scenario 2 - Dynamic Client Registration (DCR)

**Purpose**

The ability for a DH to test Dynamic Client Registration (DCR) of the CTS ADR software product.

**Scenario Conditions**

* This is a mandatory test.
* DCR steps are part of most of the scenarios.
* When successful, the ADR software product is registered with the DH. The DH must return a client ID to the CTS ADR.

**Endpoints**

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Description | Method | URI |
| Dynamic Client Registration | CTS ADR sends a DCR request to the Data Holder via the Registration Endpoint | POST | [Endpoint URI](#_Endpoints_used_in) |
| Discovery (Register) | DH requests the Discovery Document from the CTS CDR Register via the OIDC Discovery endpoint  | GET |
| JWKS (SSA) | DH requests the JWKS from the CTS CDR Register via the SSA JWKS Endpoint  | GET |
| JWKS (ADR) | DH requests the JWKS from the CTS ADR via the JWKS Endpoint | GET |
| Redirect URI  | DH calls the CTS ADR Redirect Uri endpoint to signin | GET |
| GetDataRecipientStatus | DH requests the data recipient status from the CTS Register via the Get Data Recipient Status Endpoint | GET |
| GetSoftwareProductStatus | DH requests the software product status from the CTS Register via the Get Software Product Status Endpoint | GET |
| GetDataRecipients | DH requests the data recipients from the CTS Register via the Getdatarecipients Endpoint | GET |

**Link to specs:**

<https://cdr-register.github.io/register/#dynamic-client-registration>

<https://cdr-register.github.io/register/#registration-request-using-jwt>

<https://cdr-register.github.io/register/#registration-response>

<https://cdr-register.github.io/register/#getdatarecipientsstatus>

<https://cdr-register.github.io/register/#getsoftwareproductsstatus>

<https://cdr-register.github.io/register/#getdatarecipients>

**Scenario Results**

**Pass**: you have achieved a successful DCR test when you receive a valid DCR request from the CTS ADR and supply a valid response.

**Fail**: you have failed the CTS DCR test if you received a valid DCR request from the CTS ADR but failed to supply a valid response.

### Scenario High Level Test Steps

1. Discovery Document Validation (see above)

#### Dynamic Client Registration (DCR)

1. CTS ADR requests DCR with the DH
	1. CTS ADR sends a DCR request to the DH.
	2. DH receives and verifies the CTS ADR DCR request
		1. DH calls the CTS ADR JWKS endpoint (using the jwks\_uri from the SSA).
		2. CTS ADR returns a response with an ADR JWKS.
		3. DH calls the CTS SSA JWKS endpoint to verify the SSA signature using the Register public keys.
		4. CTS Register returns a response with a Register JWKS.
2. DH responds to the CTS ADR DCR request
	1. DH registers the Software Product and returns a response to the CTS ADR .
	2. CTS ADR validates the DH DCR response.
3. DH polls the register to Get data recipient status
	1. DH sends a ‘data recipient status’ request to the CTS Register via the Get Data Recipient Status Endpoint.
	2. CDR Register returns a valid response (<https://cdr-register.github.io/register/#getdatarecipients>status )

 And/or DH polls the register to Get software product status

1. DH sends a ‘software product status’ request to the CTS Register via the Get Software Product Status Endpoint
2. CDR Register returns a valid response (<https://cdr-register.github.io/register/#getsoftwareproductsstatus> )

 And/or DH polls the register to Get data recipients

1. DH sends a ‘data recipients’ request to the CTS Register via the Getdatarecipients Endpoint.
2. CDR Register returns a valid response (<https://cdr-register.github.io/register/#getdatarecipients> )
3. DH selects to continue through the UI after at least 1 of the above Register APIs is polled



## Scenario 3 – Concurrent Consent

**Purpose**

The ability for a DH to test the Consent flow with the CTS ADR and CTS Consumer by granting two consent arrangements for a single ADR/Consumer pairing (multiple cdr arrangement ids). The DH receives a call from the CTS ADR to the Banking Get Accounts endpoint with the access token as part of the Consent.

**Scenario Conditions**

To be able to complete the Consent Revocation tests you will need to establish two “consents”. By calling authorize and token twice each. That way when you call the CTS consent revocation with a cdr arrangement id, the CTS will call back on their other one (each consent process created a new cdr arrangement id).

**Endpoints**

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Description | Method | URI |
| Authorise | CTS ADR requests authorisation with the DH via the Authorise Endpoint | GETPOST | [Endpoint URI](#_Endpoints_used_in) |
| Token | CTS ADR exchanges their code for a Token from the DH via the Token Endpoint CTS ADR exchanges their Refresh Token for an Access Token from the DH via the Token Endpoint | POST |
| Introspection | CTS ADR sends an Introspection request to the DH Token Introspection endpoint to retrieve information about a token | POST |
| Banking | [Phase 1](#_API’S)[Phase 2](#_API’S) | GET |

**Link to specs:**

<https://consumerdatastandardsaustralia.github.io/standards/#request-object>

<https://openid.net/specs/openid-connect-core-1_0.html#HybridFlowAuth>

<https://openid.net/specs/openid-connect-core-1_0.html#AuthRequestValidation>

<https://consumerdatastandardsaustralia.github.io/standards/#tokens>

<https://openid.net/specs/openid-connect-core-1_0.html#HybridIDToken>

<https://openid.net/specs/openid-connect-core-1_0.html#TokenEndpoint>

<https://openid.net/specs/openid-financial-api-part-2.html#introduction>

<https://openid.net/specs/openid-connect-core-1_0.html#IDToken>

<https://consumerdatastandardsaustralia.github.io/standards/#get-accounts>

<https://tools.ietf.org/html/rfc6749#section-4.1.2>

**Scenario Results**

**Pass**

Concurrent Consent - You have passed the concurrent consent test when you:

* authorise 2 consents on behalf of a single consumer
* return an authorisation and token response, for each consent
* You will than receive a call from the CTS ADR to the Banking API using the established consent and you must return a mock payload.

**Fail**

Concurrent Consent - You have failed the concurrent consent test when you do not:

* authorise 2 or more consents on behalf of a consumer
* return multiple Authorisation responses, Token responses (Access and Refresh Token),
* ensure that with each consent established, a new cdr arrangement id is created,
* do not redirect back to the CTS ADR
* do not return a valid response

You will also fail if the new consent arrangement is not created for the same customer. You will also fail if when you receive a call from the CTS ADR to the Banking API using the established consent and you do not return a mock payload.

### Scenario High Level Test Steps

1. Discovery Document Validation (see above)
2. Dynamic Client Registration (DCR) (see above)

#### Concurrent Consent – First Consent

1. CTS ADR requests authorisation with the DH via the Authorise Endpoint
	1. CTS ADR sends an Authorisation request to the DH via the Authorise endpoint.
	2. DH validates the CTS ADR Authorise request, verifying that the ADR Software product is registered with the DH and responds via the Redirect URI with Authorisation Code, State and encrypted and signed ID Token.
	3. CTS verifies the DH Authorise response.
2. CTS ADR sends a Token request to the DH via the Token Endpoint
	1. CTS ADR sends a Token request to the DH via the Token Endpoint, exchanging their Authorisation code for a Token.
	2. DH validates the CTS ADR Token request and returns a response with an Access Token (one-off & ongoing), a Refresh Token (ongoing only) and a cdr\_arrangment\_id.
	3. CTS verifies the DH Token response.

#### Introspection and Resources

1. CTS ADR sends an Introspection request to the DH via the Introspection Endpoint
	1. CTS ADR sends an Introspection request to the DH Token Introspection endpoint.
	2. DH validates the CTS ADR Introspection request and returns a valid response.
	3. CTS verifies the DH Token Introspection response.
2. CTS ADR calls the DH Banking Resource APIs and Customer APIs to confirm the DH has the ability to disclose phase 2 CDR data
	* 1. https://consumerdatastandardsaustralia.github.io/standards/#consumer-data-standards-banking-apis
		2. <https://consumerdatastandardsaustralia.github.io/standards/#consumer-data-standards-common-apis>
3. CTS ADR sends a request to the DH Get Customer Detail Endpoint
	1. CTS ADR sends a request, using the DH issued Access Token, to the DH via the Get Customer Detail API Endpoint.
	2. DH validates the CTS ADR request and returns a response with the mock Customer Detail payload.
	3. CTS verifies the DH Get Customer Detail Response.

#### Concurrent Consent – Second Consent

1. CTS ADR requests authorisation with the DH via the Authorise Endpoint for the same user
	1. CTS ADR sends an Authorisation request to the DH via the Authorise endpoint for the same user.
	2. DH validates the CTS ADR Authorise request, verifying that the ADR Software product is registered with the DH and responds via the Redirect URI with Authorisation Code, State and encrypted and signed ID Token.
	3. CTS verifies the DH Authorise response.
2. CTS ADR sends a Token request to the DH via the Token Endpoint
3. CTS ADR sends a Token request to the DH via the Token Endpoint, exchanging their Authorisation code for a Token.
4. DH validates the CTS ADR Token request and returns a response with an Access Token (one-off & ongoing), a Refresh Token (ongoing only) and a cdr\_arrangment\_id.
5. CTS verifies the DH Token response.
6. CTS ADR sends an Introspection request to the DH via the Introspection Endpoint
7. CTS ADR sends an Introspection request to the DH Token Introspection endpoint.
8. DH validates the CTS ADR Introspection request and returns a valid response.
9. CTS verifies the DH Token Introspection response.
10. CTS ADR sends a request to the DH Get Accounts Endpoint
11. CTS ADR sends a request, using the DH issued Access Token, to the DH via the Get Accounts API Endpoint.
12. DH validates the CTS ADR request and returns a response with the mock Account payload.
13. CTS verifies the DH Get Accounts Response.



## Scenario 4 – DH Initiated Revocation

**Purpose**

The ability for a DH to test the Withdrawal of Consent flow; verifying that a DH can send an **Arrangement** revocation request to the CTS ADR (DH to DR).

**Scenario Conditions**

A cdr\_arrangement\_Id was issued to the CTS ADR by the DH.

**Endpoints**

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Description | Method | URI |
| Arrangement Revocation DH to DR | DH sends a request, using their cdr arrangement id, to the CTS ADR to withdraw arrangement consent | POST | [Endpoint URI](#_Endpoints_used_in) |

**Link to specs:**

<https://consumerdatastandardsaustralia.github.io/standards/#end-points>

**Scenario Results**

**Pass**:

You have passed the Withdrawal of Consent flow when you call the CTS ADR Arrangement revocation endpoint and receive a success code response.

**Fail:**

You have failed the Withdrawal of Consent flow when you call the CTS ADR Arrangement revocation endpoint and **do not** receive a success code response.

### Scenario High Level Test Steps

1. Discovery Document Validation (see above)
2. Dynamic Client Registration (DCR) (see above)
3. Consent (Authorise & Token) (see above)

#### DH Initiated Revocation

1. DH sends an Arrangement Revocation request to the CTS ADR
2. DH sends an Arrangement Revocation request, using the cdr arrangement id they issued to the ADR, to the CTS ADR to withdraw Consent.
3. CTS ADR validates the request, invalidates the Consent Arrangement and returns a success code response.



## Scenario 5 – DR Initiated Revocation

**Purpose**

The ability for a DH to test the Withdrawal of Consent flow; verifying that a DH can receive an **Arrangement** revocation request from the CTS ADR (DR to DH)

**Scenario Conditions**

A cdr\_arrangement\_Id was issued to the CTS ADR by the DH

**Endpoints**

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Description | Method | URI |
| Arrangement Revocation DR to DH | CTS ADR sends a request, using their cdr arrangement id, to the DH to withdraw arrangement consent | POST | [Endpoint URI](#_Endpoints_used_in) |

**Link to specs:**

<https://consumerdatastandardsaustralia.github.io/standards/#end-points>

**Scenario Results**

**Pass**:

You have passed the Withdrawal of Consent flow when the CTS ADR calls your Arrangement revocation endpoint and receives a success code response.

**Fail:**

You have failed the Withdrawal of Consent flow when the CTS ADR calls your Arrangement revocation endpoint and **does not** receive a success code response.

### Scenario High Level Test Steps

1. Discovery Document Validation (see above)
2. Dynamic Client Registration (DCR) (see above)
3. Consent (Authorise & Token) (see above)

#### ADR Initiated Revocation

1. CTS ADR sends an Introspection request to the DH via the Introspection Endpoint
	1. CTS ADR sends an Introspection request to the DH Token Introspection endpoint.
	2. DH validates the CTS ADR Introspection request and returns a response.
	3. CTS verifies the DH Token Introspection response.
2. DH receives an Arrangement Revocation request from the CTS ADR
3. CTS ADR sends an Arrangement Revocation request to the DH Arrangements Revoke Endpoint (registered uri).
4. DH validates the request and returns a success code response.
5. CTS ADR verifies the DH Arrangement Revocation response code.
6. CTS ADR sends an Introspection request to the DH via the Introspection Endpoint
7. CTS ADR sends an Introspection request to the DH Token Introspection endpoint.
8. DH validates the CTS ADR Introspection request and returns a response.
9. CTS verifies the DH Token Introspection response.
10. CTS ADR sends a Token request to the DH with a Refresh Token
11. CTS ADR sends a Token request to the DH via the Token Endpoint, exchanging their Refresh Token for an Access Token.
12. DH validates the CTS ADR Refresh Token request and returns a response
13. CTS verifies the DH Token response [bad request].



## Scenario 6 – Removed Software Product

**Purpose**

The ability for the DH to test when an ‘active’ ADR software product changes to ’removed’ that they react accordingly.

The ACCC Registrar can change the status of a Software Product independently of the ADR accreditation status. Therefore, Data Holder's must determine the status of an ADR Software Product prior to data disclosure, consent and registration management.

**Scenario Conditions**

NA

**Endpoints**

No new endpoints are used, those used in previous scenarios are reused. Those specific to this scenario are listed below.

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Description | Method | URI |
| Banking | [Phase 1](#_API’S)[Phase 2](#_API’S) | GET | [Endpoint URI](#_Endpoints_used_in) |
| Common | [Phase 1](#_API’S)[Phase 2](#_API’S) | GET |
| GetDataRecipientStatus | DH requests the data recipient status from the CTS Register via the Get Data Recipient Status Endpoint | GET |
| GetDataRecipients | DH requests the data recipients from the CTS Register via the Getdatarecipients Endpoint | GET |
| GetSoftwareProductStatus | DH requests the software product status from the CTS Register via the Get Software Product Status Endpoint | GET |
| Arrangement Revocation DR to DH | CTS ADR sends a request, using their cdr arrangement id, to the DH to withdraw arrangement consent | POST |

**Link to specs:**

<https://cdr-register.github.io/register/#data-holder-responsibilities>

**Scenario Results**

**Pass**: You have passed the Register Status tests when an ADR Software Product status is ‘Removed’ you:

* do not disclose CDR data
* do not facilitate consent authorisation
* do not facilitate consent withdrawal.

**Fail:** You have failed the Register Status tests when an ADR Software Product status is ‘Removed’ you:

* disclose CDR data
* facilitate consent authorisation
* facilitate consent withdrawal.

### Scenario High Level Test Steps

1. Discovery Document Validation (see above)
2. Dynamic Client Registration (DCR) (see above)
3. Consent (Authorise & Token) (see above)

#### Removed Software Product

1. CTS ADR sends a request to the DH Get Banking
	1. CTS ADR sends a request, using the DH issued Access Token, to the DH via the Resource APIs
		1. https://consumerdatastandardsaustralia.github.io/standards/#consumer-data-standards-banking-apis
	2. DH validates the CTS ADR request and returns a response with the mock payloads.
	3. CTS verifies the DH Banking API Responses.
	4. CTS changes the ADR Software Product status in the CTS Register from ‘Active’ to ‘Removed’.
2. DH polls the register to Get data recipient status
3. DH sends a ‘data recipient status’ request to the CTS Register via the Get Data Recipient Status Endpoint.
4. CTS Register returns a valid response (<https://cdr-register.github.io/register/#getdatarecipients>status)

 And/or DH polls the register to Get data recipients

1. DH sends a ‘data recipients’ request to the CTS Register via the Getdatarecipients Endpoint.
2. CDR Register returns a valid response (<https://cdr-register.github.io/register/#getdatarecipients>)

And/or DH polls the register to Get software product status

1. DH sends a ‘software product status’ request to the CTS Register via the Get Software Product Status Endpoint
2. CDR Register returns a valid response (<https://cdr-register.github.io/register/#getsoftwareproductsstatus> )
3. DH selects to continue through the UI after at least 1 of the above Register APIs is polled
4. CTS confirms inability to facilitate an authorisation failure
5. CTS ADR sends an Authorisation request to the DH Authorisation Endpoint. Unlike the Authorisation request in Step 3, this is done as a backchannel request from CTS to the DH Authorisation Endpoint, with CTS impersonating a user agent. (Note - this is the only way CTS can verify that the error response is returned to the customer, rather than the ADR)
6. DH validates the request and returns a failure code response. [HTTP Status code 403 - *for more information please see the technical note under the diagram below*].
7. CTS verifies the response.
8. CTS confirms non-disclosure of CDR data
9. CTS ADR calls the DH Banking Resource APIs to confirm the DH does not disclose CDR data.
	* 1. https://consumerdatastandardsaustralia.github.io/standards/#consumer-data-standards-banking-apis
		2. <https://consumerdatastandardsaustralia.github.io/standards/#consumer-data-standards-common-apis>
10. CTS confirms inability to facilitate consent withdrawal
11. CTS ADR sends an Arrangement Revocation request to the DH Arrangements Revocation Endpoint.
12. DH validates the request and returns a failure code response [HTTP Status code 403 or 422 – *for more information please see the technical note under the diagram below]*.
13. CTS verifies the response.



### Technical note:

Where a Participant has adopted the Standardised Error Handling Requirements, CTS will validate that the participant responses conform to CDS 1.10+ requirements - which would include validation of HTTP Status Codes, Error schema and the Standard Error code itself to ensure that the Error code correlates to the specific failure condition. Where Participants have not yet transitioned to the standardised error codes, CTS validations for Error handling would not apply.

The following CDS v1.10.0+HTTP status codes and URNs will result in a pass:

|  |  |
| --- | --- |
| HTTP status code | URN |
| 403 | urn:au-cds:error:cds-all:Authorisation/AdrStatusNotActive |
| 403 | urn:au-cds:error:cds-all:Authorisation/RevokedConsent |
| 403 | urn:au-cds:error:cds-all:Authorisation/InvalidConsent |
| 422 | urn:au-cds:error:cds-all:Authorisation/InvalidArrangement |

## Scenario 7 – Reactivate Software Product

**Purpose**

The ability for the DH to test when an ‘active’ ADR software product changes to ‘inactive’ that they react accordingly. And then when the ‘Inactive ADR software product changes back to ‘active’ that the DH reacts accordingly.

The ACCC Registrar can change the status of a Software Product independently of the ADR accreditation status. Therefore, Data Holder's must determine the status of an ADR Software Product prior to data disclosure, consent and registration management.

**Scenario Conditions**

NA

**Endpoints**

No new endpoints are used, those used in previous scenarios are reused. Those specific to this scenario are listed below.

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Description | Method | URI |
| Banking | [Phase 1](#_API’S)[Phase 2](#_API’S) | GET | [Endpoint URI](#_Endpoints_used_in) |
| Common | [Phase 1](#_API’S)[Phase 2](#_API’S) | GET |
| GetDataRecipientStatus | DH requests the data recipient status from the CTS Register via the Get Data Recipient Status Endpoint | GET |
| GetDataRecipients | DH requests the data recipients from the CTS Register via the Getdatarecipients Endpoint | GET |
| GetSoftwareProductStatus | DH requests the software product status from the CTS Register via the Get Software Product Status Endpoint | GET |
| Arrangement Revocation DR to DH | CTS ADR sends a request, using their cdr arrangement id, to the DH to withdraw arrangement consent | POST |

**Link to specs:**

<https://cdr-register.github.io/register/#data-holder-responsibilities>

**Scenario Results**

**Pass**: You have passed the Register Status tests when an ADR Software Product status is ‘Inactive’ you:

* do not disclose CDR data
* do not facilitate consent authorisation
* facilitate consent withdrawal.

And when an ADR Software Product status is changes back to ‘active’ you:

* facilitate consent authorisation
* disclose CDR data

**Fail:** You have failed the Register Status tests when an ADR Software Product status is ‘Inactive’ you:

* disclose CDR data
* facilitate consent authorisation
* do not facilitate consent withdrawal.

And when an ADR Software Product status is changes back to ‘active’ you:

* do not facilitate consent authorisation
* do not disclose CDR data

### Scenario High Level Test Steps

1. Discovery Document Validation (see above)
2. Dynamic Client Registration (DCR) (see above)
3. Consent (Authorise & Token) (see above)

#### Reactivate Software Product

1. CTS ADR sends a request to the DH Get Banking/Accounts Endpoints
	1. CTS ADR sends a request, using the DH issued Access Token, to the DH via the Resource APIs
		1. https://consumerdatastandardsaustralia.github.io/standards/#consumer-data-standards-banking-apis
		2. <https://consumerdatastandardsaustralia.github.io/standards/#consumer-data-standards-common-apis>
	2. DH validates the CTS ADR request and returns a response with the mock payloads.
	3. CTS verifies the DH Banking API Responses.
	4. CTS changes the ADR Software Product status in the CTS Register from ‘Active’ to ‘Inactive.
2. DH polls the register to Get data recipient status
3. DH sends a ‘data recipient status’ request to the CTS Register via the Get Data Recipient Status Endpoint.
4. CTS Register returns a valid response (<https://cdr-register.github.io/register/#getdatarecipients>status)

 DH polls the register to Get data recipients

1. DH sends a ‘data recipients’ request to the CTS Register via the Getdatarecipients Endpoint.
2. CDR Register returns a valid response (<https://cdr-register.github.io/register/#getdatarecipients>)

And/or DH polls the register to Get software product status

1. DH sends a ‘software product status’ request to the CTS Register via the Get Software Product Status Endpoint
2. CDR Register returns a valid response (<https://cdr-register.github.io/register/#getsoftwareproductsstatus> )
3. DH selects to continue through the UI after at least 1 of the above Register APIs is polled
4. CTS confirms non-disclosure of CDR data
5. CTS ADR calls the DH Banking Resource APIs to confirm the DH does not disclose CDR data.
6. https://consumerdatastandardsaustralia.github.io/standards/#consumer-data-standards-banking-apis
7. <https://consumerdatastandardsaustralia.github.io/standards/#consumer-data-standards-common-apis>
8. CTS confirms inability to facilitate an authorisation request
9. CTS ADR sends an Authorisation request to the DH Authorisation Endpoint. Unlike the Authorisation request in Step 3 and Step 12, this is done as a backchannel request from CTS to the DH Authorisation Endpoint, with CTS impersonating a user agent. (Note - this is the only way CTS can verify that the error response is returned to the customer, rather than the ADR)
10. DH validates the request and returns a failure code response [HTTP Status code 403 - *for more information please see the technical note under the diagram below*]
11. CTS verifies the response.
12. CTS confirms ability to facilitate consent withdrawal
13. CTS ADR sends an Arrangement Revocation request to the DH Arrangements Revoke Endpoint.
14. DH validates the request and returns a success code response.
15. CTS verifies the response.
16. CTS changes the ADR Software Product status in the CTS Register from ‘Inactive’ to ‘Active’.
17. DH polls the register to Get data recipient status
18. DH sends a ‘data recipient status’ request to the CTS Register via the Get Data Recipient Status Endpoint.
19. CTS Register returns a valid response (<https://cdr-register.github.io/register/#getdatarecipients>status)

 DH polls the register to Get data recipients

1. DH sends a ‘data recipients’ request to the CTS Register via the Getdatarecipients Endpoint.
2. CDR Register returns a valid response (<https://cdr-register.github.io/register/#getdatarecipients>)
3. DH selects to continue through the UI after at least 1 of the above Register APIs is polled
4. Consent (Authorise & Token) (see above)
5. CTS ADR sends a request to the DH Get Accounts Endpoint
6. CTS ADR sends a request, using the DH issued Access Token, to the DH via the Get Accounts API Endpoint.
7. DH validates the CTS ADR request and returns a response with the mock Account payload.
8. CTS verifies the DH Get Accounts Response.

 

### Technical note:

Where a Participant has adopted the Standardised Error Handling Requirements, CTS will validate that the participant responses conform to CDS 1.10+ requirements - which would include validation of HTTP Status Codes, Error schema and the Standard Error code itself to ensure that the Error code correlates to the specific failure condition. Where Participants have not yet transitioned to the standardised error codes, CTS validations for Error handling would not apply.

The following status codes and URNs will result in a pass:

|  |  |  |
| --- | --- | --- |
| CDS Version | HTTP status code | URN |
| 1.9 | 4xx | N/A |
| 1.10.0 | 403 | urn:au-cds:error:cds-all:Authorisation/AdrStatusNotActive |
| 1.10.0 | 403 | **meta object:**urn:au-cds:error:cds-all:Authorisation/AdrStatusNotActive |

## Scenario 8 – Replace Existing Consent with PAR

**Purpose**

The ability for a DH to test the replacement or extension of an existing Consent Arrangement with the CTS ADR using the Pushed Authorisation Request (PAR) endpoint.

**Scenario Conditions**

NA

**Endpoints**

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Description | Method | URI |
| PAR | DH receives a request with a Client Assertion via their PAR Endpoint to extend or replace an existing Consent Arrangement  | POST | [Endpoint URI](#_Endpoints_used_in) |

**Link to specs:**

<https://consumerdatastandardsaustralia.github.io/standards/#end-points>

**Scenario Results**

**Pass**:

You have passed the Pushed Authorisation Request (PAR) flow when the CTS ADR sends you a PAR request and receives a valid response. The CTS ADR will then make an authorisation request (using the request\_uri from the PAR response), followed by a token request. You will than receive a call from the CTS ADR to the Banking API using the established consent and you must return a mock payload. The CTS ADR will then call your Token endpoint with a Refresh Token from the replaced Consent Arrangement and expect to receive a response with an error code.

**Fail:**

You have failed the Pushed Authorisation Request flow when the CTS ADR **cannot** send you a PAR request and/or receives an **invalid** response. The CTS ADR will then make an authorisation request (using the request\_uri from the PAR response), followed by a token request. You will than receive a call from the CTS ADR to the Banking API using the established consent and you **do not** return a mock payload. The CTS ADR will then call your Token endpoint with a Refresh Token from the replaced Consent Arrangement and does not receive a response with an error code.

### Scenario High Level Test Steps

1. Discovery Document Validation (see above)
2. Dynamic Client Registration (DCR) (see above)
3. Consent (Authorise & Token) (see above)
4. CTS ADR sends a request to the DH Get Banking/Accounts Endpoint
	1. CTS ADR sends a request, using the DH issued Access Token, to the DH via the Get Accounts API Endpoint.
	2. DH validates the CTS ADR request and returns a response with the mock Account payload.
	3. CTS verifies the DH Get Accounts Response.

**PAR**

1. CTS ADR sends a Pushed Authorisation Request (PAR) to the DH
2. CTS ADR sends a PAR request, with Client Authentication and replacement Claims, to the DH via the PAR Endpoint.
3. DH validates the PAR request
4. DH returns a response to the PAR request with a RequestURI and ExpiresIn.
5. CTS ADR validates the DH PAR response.
6. Replace Consent (Authorise & Token) (see above)
7. CTS ADR sends a request to the DH Get Banking/Accounts API
8. CTS ADR sends a request, using the DH issued Access Token, to the DH via the Get Accounts API Endpoint.
9. DH validates the CTS ADR request and returns a response with the mock Account payload.
10. CTS verifies the DH Get Accounts Response.
11. CTS ADR sends a Token request to the DH with a Refresh Token to replace the Consent
12. CTS ADR sends a Token request to the DH via the Token Endpoint, exchanging their Refresh Token of the original consent request for an Access Token.
13. DH validates the CTS ADR Refresh Token request and returns a response.
14. CTS verifies the DH Token response [bad request].



## Scenario 9 – Register PUT GET

**Purpose**

The ability for a DH to test they can respond to a PUT and GET DCR request for a given Client ID.

**Scenario Conditions**

NA

**Endpoints**

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Description | Method | URI |
| Dynamic Client Registration | CTS ADR sends a DCR request to the Data Holder via the Register Endpoint | PUTGET | [Endpoint URI](#_Endpoints_used_in) |

**Link to specs:**

<https://cdr-register.github.io/register/#register-a-client-using-a-cdr-register-issued-software-statement-assertion>

<https://cdr-register.github.io/register/#registration-request-using-jwt>

<https://cdr-register.github.io/register/#registration-response>

<https://cdr-register.github.io/register/#dynamic-client-registration>

**Scenario Results**

**Pass**:

You have passed the Register PUT GET Scenario if you can process both a GET and a PUT to the registration endpoint.

**Fail:**

You have failed the Register PUT GET Scenario if you cannot process both a GET and a PUT to the registration endpoint.

### Scenario High Level Test Steps

1. Discovery Document Validation (see above)
2. Dynamic Client Registration (DCR) (see above)
3. Consent (Authorise & Token) (see above)

**Registry Put Get**

1. CTS sends a POST Token Request to the DH
2. CTS sends a POST Token request to the DH via the Token Endpoint using ClientCredentials grant type
3. DH validates the CTS request and returns a response
4. CTS verifies the response
5. CTS ADR sends a PUT Register request to the DH Client Registration (DCR)
6. CTS ADR sends a PUT Registration Request containing the bearer token from step 4, with an updated SSA to reflect the changes, to the DH via the /register (DCR) Endpoint.
7. DH validates the CTS request and returns a response
8. CTS verifies the response
9. CTS ADR sends a GET Register request to the DH Client Registration (DCR)
10. CTS ADR sends a GET Register request with the bearer token to the DH via the /register (DCR) Endpoint.
11. DH validates the CTS request and returns a response.
12. CTS verifies the response equates to the request that was sent to the GET/register endpoint.



## Scenario 10 – Token Revocation

**Purpose**

The ability for a DH to test the Withdrawal of a Token verifying that a DH can receive a **Token** revocation request from the CTS ADR (DR to DH)

**Scenario Conditions**

NA

**Endpoints**

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Description | Method | URI |
| Token Revocation DR to DH | CTS ADR sends a Token Revocation request, using their token, to the DH to withdraw a token | POST | [Endpoint URI](#_Endpoints_used_in) |

**Link to specs:**

<https://consumerdatastandardsaustralia.github.io/standards/#end-points>

**Scenario Results**

**Pass**:

You have passed the Withdrawal of Token flow when you receive a Token Revocation request from the CTS ADR to the DH Revocation Endpoint with a Refresh Token**.** You validate the request and returns a success code (200 OK) response. You will than receive a call from the CTS ADR to the Banking API using the established consent and you must return a mock payload. The CTS ADR makes a second identical Token Revocation request to the DH Revocation Endpoint. You validate the request.

**Fail:**

You have failed the Withdrawal of Token flow when you **cannot** receive a Token Revocation request from the CTS ADR to the DH Revocation Endpoint with a Refresh Token, or you **do not** validate the request or return a success code (200 OK) response. You will than receive a call from the CTS ADR to the Banking API using the established consent and you **do not** return a mock payload. The CTS ADR will make a second identical Token Revocation request to the DH Revocation Endpoint. You have failed if you do notvalidate the request or return a failure code (not a 200 OK) response.

### Scenario High Level Test Steps

1. Discovery Document Validation (see above)
2. Dynamic Client Registration (DCR) (see above)
3. Consent (Authorise & Token) (see above)

**Token Revocation DR to DH**

1. CTS ADR sends a Token Revocation Request (Access Token) to the DH
2. CTS ADR sends a Token Revocation request, with the Access Token to the DH via the Revocation Endpoint.
3. DH validates the CTS ADR request and returns a response.
4. CTS ADR verifies the response.
5. CTS ADR sends a request to the DH Get Banking/Accounts API
6. CTS ADR sends a request, using the DH issued Access Token, to the DH via the Get Accounts API Endpoint.
7. DH validates the CTS ADR request and returns a response.
8. CTS verifies the DH Get Accounts Response [bad request].
9. CTS ADR sends a Token Revocation Request (Refresh Token) to the DH
10. CTS ADR sends a Token Revocation request, with the Refresh Token to the DH via the Revocation Endpoint.
11. DH validates the CTS ADR request and returns a response.
12. CTS ADR verifies the response.
13. CTS ADR sends a Token Request (Refresh Token) to the DH
14. CTS ADR sends a Token request, with the Refresh Token to the DH via the Token Endpoint.
15. CTS ADR verifies the response [bad request].



## Scenario 11a – Get Software Product Status Register Polling

**Purpose**

The ability for a DH to test their ability to react to ADRs and their associated Software Statuses changes within 5 minutes of the change occurring on the CDR Register for Software Product Satus. To achieve this, DH can poll the GetSoftwareProductsStatus APIs to retrieve the current statuses and cache these for use during requests for Consumer Data.

**Scenario Conditions**

NA

**Endpoints**

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Description | Method | URI |
| GetSoftwareProductStatus | DH requests the software product status from the CTS Register via the Get Software Product Status Endpoint | GET | [Endpoint URI](#_Endpoints_used_in) |

**Link to specs:**

<https://consumerdatastandardsaustralia.github.io/standards/#end-points>

<https://cdr-register.github.io/register/#getsoftwareproductstatus>

**Scenario Results**

**Pass**: You have passed the CTS CDR Register Polling tests when you call the Software Product Status endpoint twice within 5 minutes.

**Fail**: You have failed the CTS CDR Register Polling tests when you **do not** call the Software Product Status endpoint twice within 5 minutes.

### Scenario High Level Test Steps

1. Discovery Document Validation (see above)
2. Dynamic Client Registration (DCR) (see above)

**5 Minute Polling Check – Get Software Product Status**

1. DH polls the register to Get software product status FIRST CALL
2. DH sends a ‘software product status’ request to the CTS Register via the getsoftwareproductsstatus endpoint.
3. CDR Register returns a valid response (<https://cdr-register.github.io/register/#getsoftwareproductsstatus>).
4. CTS captures a timestamp of first polling request.
5. DH polls the register to Get software product status SECOND CALL
6. DH sends a ‘software product status’ request to the CTS Register via the getsoftwareproductsstatus endpoint.
7. CDR Register returns a valid response (<https://cdr-register.github.io/register/#getsoftwareproductsstatus>).
8. CTS captures a timestamp of second polling request.
9. DH selects to continue execution of the Scenario through the UI
10. CTS verifies inbound status request time difference
11. CTS compares the timestamps of the two polling requests.
12. CTS verifies a second status request was received in less than 5 minutes.



## Scenario 11b – Get Data Recipients Register Polling

**Purpose**

The ability for a DH to test their ability to react to ADRs and their associated Software Statuses changes within 5 minutes of the change occurring on the CDR Register for Data Recipients. To achieve this, DH can poll the GetDataRecipients APIs to retrieve the current statuses and cache these for use during requests for Consumer Data.

**Scenario Conditions**

NA

**Endpoints**

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Description | Method | URI |
| GetDataRecipients | DH requests the data recipients from the CTS Register via the Get Data Recipient Status Endpoint | GET | [Endpoint URI](#_Endpoints_used_in) |

**Link to specs:**

<https://consumerdatastandardsaustralia.github.io/standards/#end-points>

<https://cdr-register.github.io/register/#getdatarecipients>

**Scenario Results**

**Pass**: You have passed the CTS CDR Register Polling tests when you call the Data Recipients endpoint twice within 5 minutes.

**Fail**: You have failed the CTS CDR Register Polling tests when you **do not** call the Data Recipients endpoint twice within 5 minutes.

### Scenario High Level Test Steps

1. Discovery Document Validation (see above)
2. Dynamic Client Registration (DCR) (see above)

**5 Minute Polling Check – Get Data Recipients**

1. DH polls the register to Get data recipients FIRST CALL
2. DH sends a ‘data recipients’ request to the CTS Register via the Getdatarecipients Endpoint.
3. CDR Register returns a valid response (<https://cdr-register.github.io/register/#getdatarecipients>).
4. CTS captures a timestamp of first polling request.
5. DH polls the register to Get data recipients SECOND CALL
6. DH sends a ‘data recipients’ request to the CTS Register via the Getdatarecipients Endpoint.
7. CDR Register returns a valid response (<https://cdr-register.github.io/register/#getdatarecipients>).
8. CTS captures a timestamp of second polling request.
9. DH selects to continue execution of the Scenario through the UI
10. CTS verifies inbound status request time difference
11. CTS compares the timestamps of the two polling requests.
12. CTS verifies a second status request was received in less than 5 minutes.



## Scenario 12 – Ensure Client Assertion Data in Token Request

**Purpose**

The ability for a DH to complete negative tests for the Client Assertion (Authentication) via the Token endpoint with the CTS ADR. To achieve this the CTS ADR will send multiple bad requests to the DHs Token endpoint.

**Scenario Conditions**

NA

**Endpoints**

No new endpoints are used, those used in previous scenarios are reused. Those specific to this scenario are listed below.

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Description | Method | URI |
| Token | CTS ADR exchanges their code for a Token from the DH via the Token Endpoint  | POST | [Endpoint URI](#_Endpoints_used_in) |

**Link to specs:**

<https://consumerdatastandardsaustralia.github.io/standards/#end-points>

<https://consumerdatastandardsaustralia.github.io/standards/#request-object>

<https://openid.net/specs/openid-connect-core-1_0.html#HybridFlowAuth>

<https://consumerdatastandardsaustralia.github.io/standards/#tokens>

<https://openid.net/specs/openid-connect-core-1_0.html#HybridIDToken>

<https://openid.net/specs/openid-connect-core-1_0.html#TokenEndpoint>

**Scenario Results**

**Pass**: You have passed the ‘Client Assertion Data in Token Request’ tests when you handle each negative test correctly.

**Fail**: You have failed the CTS Client Assertion Data in Token Request tests when you **don’t** handle each negative test correctly.

### Scenario High Level Test Steps

1. Discovery Document Validation (see above)
2. Dynamic Client Registration (DCR) (see above)
3. CTS ADR requests authorisation with the DH via the Authorise Endpoint
	1. CTS ADR sends an Authorisation request to the DH via the Authorise endpoint.
	2. DH validates the CTS ADR Authorise request, verifying that the ADR Software product is registered with the DH and responds via the Redirect URI with Authorisation Code, State and encrypted and signed ID Token.
	3. CTS verifies the DH Authorise response.

**Ensure Client Assertion Data in Token Request**

1. CTS ADR sends a Token request to the DH via the Token Endpoint
	1. CTS ADR sends a Token request **without a client assertion** to the DH via the Token Endpoint, exchanging their Authorisation code for a Token.
	2. DH validates the CTS ADR Token request and returns a response
2. CTS verifies the DH Token response [bad request]
3. CTS ADR sends a Token request to the DH via the Token Endpoint
	1. CTS ADR sends a Token request with a **wrong ‘aud’** in client assertion to the DH via the Token Endpoint, exchanging their Authorisation code for a Token.
	2. DH validates the CTS ADR Token request and returns a response
	3. CTS verifies the DH Token response [bad request]
4. CTS ADR sends a Token request to the DH via the Token Endpoint
	1. CTS ADR sends a Token request **missing ‘aud’** in client assertion to the DH via the Token Endpoint, exchanging their Authorisation code for a Token.
	2. DH validates the CTS ADR Token request and returns a response
	3. CTS verifies the DH Token response [bad request]
5. CTS ADR sends a Token request to the DH via the Token Endpoint
	1. CTS ADR sends a Token request **wrong ‘sub’** in client assertion to the DH via the Token Endpoint, exchanging their Authorisation code for a Token.
	2. DH validates the CTS ADR Token request and returns a response
	3. CTS verifies the DH Token response [bad request]
6. CTS ADR sends a Token request to the DH via the Token Endpoint
	1. CTS ADR sends a Token request with a **missing ‘sub’** in client assertion to the DH via the Token Endpoint, exchanging their Authorisation code for a Token.
	2. DH validates the CTS ADR Token request and returns a response
	3. CTS verifies the DH Token response [bad request]
7. CTS ADR sends a Token request to the DH via the Token Endpoint
	1. CTS ADR sends a Token request with **an expired** client assertion to the DH via the Token Endpoint, exchanging their Authorisation code for a Token.
	2. DH validates the CTS ADR Token request and returns a response
	3. CTS verifies the DH Token response [bad request]
8. CTS ADR sends a Token request to the DH via the Token Endpoint
	1. CTS ADR sends a Token request with a **missing ‘iss’** in client assertion to the DH via the Token Endpoint, exchanging their Authorisation code for a Token.
	2. DH validates the CTS ADR Token request and returns a response
	3. CTS verifies the DH Token response [bad request]
9. CTS ADR sends a Token request to the DH via the Token Endpoint
	1. CTS ADR sends a Token request with a **wrong ‘iss’** in client assertion to the DH via the Token Endpoint, exchanging their Authorisation code for a Token.
	2. DH validates the CTS ADR Token request and returns a response
	3. CTS verifies the DH Token response [bad request]
10. CTS ADR sends a Token request to the DH via the Token Endpoint
	1. CTS ADR sends a Token request with a **missing ‘alg’** in client assertion to the DH via the Token Endpoint, exchanging their Authorisation code for a Token.
	2. DH validates the CTS ADR Token request and returns a response
	3. CTS verifies the DH Token response [bad request]
11. CTS ADR sends a Token request to the DH via the Token Endpoint
	1. CTS ADR sends a Token request with an **empty ‘alg’** in client assertion to the DH via the Token Endpoint, exchanging their Authorisation code for a Token.
	2. DH validates the CTS ADR Token request and returns a response
	3. CTS verifies the DH Token response [bad request]
12. CTS ADR sends a Token request to the DH via the Token Endpoint
	1. CTS ADR sends a Token request with a **‘none’ ‘alg’** in client assertion to the DH via the Token Endpoint, exchanging their Authorisation code for a Token.
	2. DH validates the CTS ADR Token request and returns a response
	3. CTS verifies the DH Token response [bad request]
13. CTS ADR sends a Token request to the DH via the Token Endpoint
	1. CTS ADR sends a Token request with a **‘RS256’ ‘alg’** in client assertion to the DH via the Token Endpoint, exchanging their Authorisation code for a Token.
	2. DH validates the CTS ADR Token request and returns a response
	3. CTS verifies the DH Token response [bad request]
14. CTS ADR sends a Token request to the DH via the Token Endpoint
	1. CTS ADR sends a Token request with a **wrong signature** in client assertion to the DH via the Token Endpoint, exchanging their Authorisation code for a Token.
	2. DH validates the CTS ADR Token request and returns a response
	3. CTS verifies the DH Token response [bad request]
15. CTS ADR sends a Token request to the DH via the Token Endpoint
	1. CTS ADR sends a Token request with a **missing ‘client\_assertion\_type’** to the DH via the Token Endpoint, exchanging their Authorisation code for a Token.
	2. DH validates the CTS ADR Token request and returns a response
	3. CTS verifies the DH Token response [bad request]
16. CTS ADR sends a Token request to the DH via the Token Endpoint
	1. CTS ADR sends a Token request with a **wrong ‘client\_assertion\_type’** to the DH via the Token Endpoint, exchanging their Authorisation code for a Token.
	2. DH validates the CTS ADR Token request and returns a response
	3. CTS verifies the DH Token response [bad request]





## Scenario 13 – Amending Account for an Existing Consent Scenario with PAR

**Purpose**

The ability for a DH to test the amendment of an existing Consent Arrangement with the CTS ADR using the Pushed Authorisation Request (PAR) endpoint to enable a new sharing duration, scope and account from the existing (initial) PAR request.

**Scenario Conditions**

NA

**Endpoints**

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Description | Method | URI |
| PAR | DH receives a request with a Client Assertion via their PAR Endpoint to amend an existing Consent Arrangement  | POST | [Endpoint URI](#_Endpoints_used_in) |

**Link to specs:**

<https://consumerdatastandardsaustralia.github.io/standards/#end-points>

**Scenario Results**

**Pass**:

You have passed the ‘Amending Account for an Existing Consent Scenario’ with PAR flow when the CTS ADR sends you a PAR request and receives a valid response. The CTS ADR will then make an authorisation request (using the request\_uri from the PAR response), followed by a token request. The token response must contain the amended sharing duration and scopes from initial PAR request

You will than receive a call from the CTS ADR to the Get Accounts API using the established consent and you must return a mock payload. The returned payload is validated against the initial Get Accounts response and must not be equal to the previous captured accounts.

The CTS ADR will then call your Token endpoint with a Refresh Token from the replaced Consent Arrangement and expect to receive a response with an error code.

**Fail:**

You have failed the Amending Account for an Existing Consent Scenario with PAR flow when the CTS ADR **cannot** send you a PAR request and/or receives an **invalid** response. The CTS ADR will then make an authorisation request (using the request\_uri from the PAR response), followed by a token request. The token response **does not** contain the amended sharing duration and scopes from initial PAR request.

You will then receive a call from the CTS ADR to the Get Accounts API using the established consent and you **do not** return a mock payload. You have failed if a mock payload is returned it is validated against the initial Get Accounts response **and is equal** to the previous captured accounts.

The CTS ADR will then call your Token endpoint with a Refresh Token from the replaced Consent Arrangement and does not receive a response with an error code.

### Scenario High Level Test Steps

1. Discovery Document Validation (see above)
2. Dynamic Client Registration (DCR) (see above)
3. Consent (Authorise & Token) (see above)
4. CTS ADR sends a request to the DH Get Accounts Endpoint
	1. CTS ADR sends a request, using the DH issued Access Token, to the DH via the Get Accounts API Endpoint.
	2. DH validates the CTS ADR request and returns a response with the mock Account payload.
	3. CTS verifies the DH Get Accounts Response.

**Amend Account with PAR**

1. CTS ADR sends a Pushed Authorisation Request (PAR) to the DH
2. CTS ADR sends a PAR request, with Client Authentication and amended Claims, to the DH via the PAR Endpoint.
3. DH validates the PAR request
4. DH returns a response to the PAR request with a ‘RequestURI’ and ‘ExpiresIn’.
5. CTS ADR validates the DH PAR response.
6. Replace Consent (Authorise & Token) **(**see above, **make sure you select a different account)**
7. CTS ADR sends a request to the DH Get Accounts API
8. CTS ADR sends a request, using the DH issued Access Token, to the DH via the Get Accounts API Endpoint.
9. DH validates the CTS ADR request and returns a response with the mock Account payload.
10. CTS verifies the DH Get Accounts Response.
11. CTS ADR sends a Token request to the DH with a Refresh Token to replace the Consent
12. CTS ADR sends a Token request to the DH via the Token Endpoint, exchanging their Refresh Token of the original consent request for an Access Token.
13. DH validates the CTS ADR Refresh Token request and returns a response.
14. CTS verifies the DH Token response [bad request].



## Scenario 14 – Ensure Holder of Key (HoK) for Resource Requests

**Purpose**

The ability for a DH to test their compliance to the HoK mechanism (as defined in the CDS) when accessing resource APIs.

**Scenario Conditions**

The Resource API request is made with a CDR certificate that is not bound to the access token used.

**Endpoints**

No new endpoints are used, those used in previous scenarios are reused. Those specific to this scenario are listed below.

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Description | Method | URI |
| Banking | [Phase 1](#_API’S)[Phase 2](#_API’S) | GET | [Endpoint URI](#_Endpoints_used_in) |

**Link to specs:**

<https://consumerdatastandardsaustralia.github.io/standards/#end-points> <https://consumerdatastandardsaustralia.github.io/standards/#get-accounts>

**Scenario Results**

**Pass**: You have passed the scenario when the Get Accounts request is sent with a CDR certificate and an access token which is not bound to it and the response is an error code.

**Fail:** You have failed the scenario when the Get Accounts request is sent with a CDR certificate and an access token which is not bound to it and the response returns the request of the accounts and/or it doesn’t return an error code.

### Scenario High Level Test Steps

1. Discovery Document Validation (see above)
2. Dynamic Client Registration (DCR) (see above)
3. Consent (Authorise & Token) (see above)
4. CTS ADR sends a request to the DH Get Accounts Endpoint
	1. CTS ADR sends a request to the DH Get Accounts Endpoint using the DH issued Access Token and a different CDR certificate than the one used to request the access token.
	2. DH validates the CTS ADR request and returns an error response of error code.
	3. CTS verifies the DH Get Accounts Response.



## Scenario 15 – Ensure Infosec Endpoints Using MTLS Authentication with X509 Certificates

**Purpose**

The ability for a DH to test the Infosec Endpoints that use MTLS with X509 Certificates and ensure a valid response.

**Scenario Conditions**

The Registration Request is sent without a CDR client certificate.

The Registration Request is sent with an expired CDR client certificate.

The Registration Request is sent with a non-CDR (self-signed) client certificate.

The Registration Request is sent with a revoked CDR client certificate.

**Endpoints**

No new endpoints are used, those used in previous scenarios are reused. Those specific to this scenario are listed below.

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Description | Method | URI |
| Dynamic Client Registration | CTS ADR sends a DCR request to the Data Holder via the Registration Endpoint | POST | [Endpoint URI](#_Endpoints_used_in) |

**Link to specs:**

<https://consumerdatastandardsaustralia.github.io/standards/#end-points>

<https://cdr-register.github.io/register/#dynamic-client-registration>

<https://cdr-register.github.io/register/#registration-response>

**Scenario Results**

**Pass**: You have passed the scenario when the Registration Request is sent with an invalid/missing client certificate and the response is an appropriate error.

**Fail:** You have failed the scenario when the Registration Request is sent with an invalid/missing client certificate and the response is not an appropriate error.

### Scenario High Level Test Steps

1. Discovery Document Validation (see above)
2. CTS ADR sends a Registration Request to the DH Registration Endpoint without CDR Client Certificate
	1. CTS ADR sends a registration request without a CDR Client Certificate to the DH.
	2. SSL exception is expected, and SSL exception has occurred.
3. CTS ADR sends a Registration request to the DH Registration Endpoint with an expired CDR Client Certificate
	1. CTS ADR sends a registration request with an expired CDR Client Certificate to the DH.
	2. SSL exception is expected, and SSL exception has occurred.
4. CTS ADR sends a Registration request to the DH Registration Endpoint with a non-CDR (self-signed) Client Certificate
	1. CTS ADR sends a registration request with a non-CDR (self-signed) Client Certificate to the DH.
	2. SSL exception is expected, and SSL exception has occurred.
5. CTS ADR sends a Registration request to the DH Registration Endpoint with a revoked CDR Client Certificate
	1. CTS ADR sends a registration request with a revoked CDR Client Certificate to the DH.
	2. SSL exception is expected, and SSL exception has occurred.

 

## Scenario 16 – Consent Software Statement Assertion with Sector Identifier URI Scenario

**Purpose**

For a DH to perform a registration and consent when the optional field ‘sector\_identifier\_uri’ was present in the SSA and the ‘sub’ claim in the id token are different if ‘sector\_identifier\_uri’ is different.

**Scenario Conditions**

* This is a mandatory test.
* This is done at the beginning of a testing cycle.

**Endpoints**

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Description | Method | Uri |
| Registration Endpoint  | CTS ADR sends a DCR request to the Data Holder via the Registration Endpoint | POST | [Endpoint URI](#_Endpoints_used_in) |
| sector\_identifier\_uri | URL string referencing the CTS ADR sector identifier URI | GET |
| Redirect URI  | DH calls the CTS ADR Redirect Uri endpoint to signin | GET |

**Link to specs**

<https://consumerdatastandardsaustralia.github.io/standards/#end-points> [Dynamic Client Registration – CDR Register Design Reference (cdr-register.github.io)](https://cdr-register.github.io/register/#dynamic-client-registration)

[Registration Response – CDR Register Design Reference (cdr-register.github.io)](https://cdr-register.github.io/register/#registration-response)

<https://openid.net/specs/openid-connect-core-1_0.html#SubjectIDTypes>

**Scenario Results**

**Pass**: you have passed the scenario when

* the DH response contains the same sector\_identifier\_uri as provided in the registration request.
* the DH calls the sector\_identifier\_uri as part of the client registration
* when the sector\_identifier\_uri is different for each client then the sub claim in the id token is different as well.

**Pass with warning**: the registration response does not contain the same sector\_identifier\_uri as was provided in the registration request or is empty or the field is missing.

**Fail**: you have failed when

* the DH does not call the sector\_identifier\_uri as part of the client registration
* the sub claim in the id token is the same.

### Scenario High Level Test Steps

1. Discovery Document Validation (see above)
2. CTS ADR requests Registration with the DH for Client 1
	1. CTS ADR sends a Registration request with sector\_identifier\_uri to the DH.
	2. DH receives and verifies the CTS ADR DCR request
		1. DH calls the CTS ADR JWKS endpoint (using the jwks\_uri from the SSA).
		2. DH calls the CTS Register JWKS endpoint to verify the SSA signature using the Register public keys.
		3. DH calls the CTS ADR sector\_identifier\_uri endpoint using the sector\_identifier\_uri from the SSA.
3. DH responds to the CTS ADR Registration request for Client 1
	1. DH registers the Software Product and returns a response to the CTS ADR.
	2. CTS ADR validates the DH Registration response.
4. CTS ADR requests Registration with the DH for Client 2
	1. CTS ADR sends a registration request with sector\_identifier\_uri to the DH.
	2. DH receives and verifies the CTS ADR DCR post request
		1. DH calls the CTS ADR JWKS endpoint (using the jwks\_uri from the SSA with sector\_identifier\_uri).
		2. DH calls the CTS Register JWKS endpoint to verify the SSA signature using the Register public keys.
		3. DH calls the CTS ADR sector\_identifier\_uri endpoint using the sector\_identifier\_uri from the SSA.
5. DH responds to the CTS ADR Registration request for Client 2
	1. DH registers the Software Product and returns a response to the CTS ADR.
	2. CTS ADR validates the DH DCR response.
6. DH polls the register to Get data recipient status for Client
	1. DH sends a ‘data recipient status’ request to the CTS Register via the Get Data Recipient Status Endpoint.
	2. CDR Register returns a valid response (<https://cdr-register.github.io/register/#getdatarecipients>status )

 And/or DH polls the register to Get software product status

1. DH sends a ‘software product status’ request to the CTS Register via the Get Software Product Status Endpoint
2. CDR Register returns a valid response (<https://cdr-register.github.io/register/#getsoftwareproductsstatus> )

 And/or DH polls the register to Get data recipients

1. DH sends a ‘data recipients’ request to the CTS Register via the Getdatarecipients Endpoint.
2. CDR Register returns a valid response (<https://cdr-register.github.io/register/#getdatarecipients> )
3. CTS ADR requests authorisation with the DH via the Authorise Endpoint for Client 1
	1. CTS ADR sends an Authorisation request to the DH via the Authorise endpoint.
	2. DH validates the CTS ADR Authorise request, verifying that the ADR Software product is registered with the DH and responds via the Redirect URI with Authorisation Code, State and encrypted and signed ID Token.
	3. CTS verifies the DH Authorise response.
4. CTS ADR requests authorisation with the DH via the Authorise Endpoint for Client 2
	1. CTS ADR sends an Authorisation request to the DH via the Authorise endpoint.
	2. DH validates the CTS ADR Authorise request, verifying that the ADR Software product is registered with the DH and responds via the Redirect URI with Authorisation Code, State and encrypted and signed ID Token.
	3. CTS verifies the DH Authorise response.
5. Verification that sub claim for Client 1 and 2 are different
	1. Verify that sub claim for Client 1 and 2 are different.



# API’S

The different Phases for API types and the associated api’s.

|  |  |  |
| --- | --- | --- |
| PHASE | API TYPE | API |
| Phase 1 | Banking | Get Accounts |
| Get Bulk Balances |
| Get Balances for Specific Accounts |
| Get Account Balance |
| Get Transactions for Account |
| Get Transaction Detail |
| Common | Get Customer |
| Phase 2 also includes Phase 1. | Banking | Get Account Detail |
| Get Direct Debits for Account |
| Get Bulk Direct Debits |
| Get Direct Debits for Specific Accounts |
| Get Scheduled Payment for Account |
| Get Scheduled Payments Bulk |
| Get Scheduled Payments for Specific Accounts |
| Get Payees |
| Get Payee Detail |
| Common | Get Customer Detail |

# Endpoints used in DH Scenarios

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Endpoint | Description | URL |
| Discovery | **Discovery (DH)** | CTS ADR requests the discovery document from the Data Holder via the Discovery Endpoint | <https://www.dataholder.com.au/.well-known/openid-configuration> (Data Holder endpoint) |
| DCR | **Dynamic Client Registration** | CTS ADR sends a DCR request to the Data Holder via the Registration Endpoint | [https://www.dataholder.com.au/register](https://www.dataholder.com.au/register%C2%A0)(Data Holder endpoint) |
|  | **Discovery (Register)** | DH requests the Discovery Document from the CTS CDR Register via the OIDC Discovery endpoint  | https://api.cts.cdr.gov.au/{conformanceId}/register/idp/.well-known/openid-configuration |
|  | **JWKS (Register)** | DH requests the JWKS from the CTS CDR Register via the JWKS Endpoint  | [https://api.cts.cdr.gov.au/cts/{conformanceId}/register/cdr-register/v1/jwks](https://api.cts.cdr.gov.au/cts/%7BconformanceId%7D/register/cdr-register/v1/jwks)  |
|  | **JWKS (ADR)** | DH requests the JWKS from the CTS ADR via the JWKS Endpoint | [https://api.cts.cdr.gov.au/cts/{conformanceId-guid}/dr/jwks](https://api.cts.cdr.gov.au/cts/%7BconformanceId-guid%7D/dr/jwks) |
|  | **URI signin** | DH calls the CTS ADR Redirect Uri endpoint to signin | [https://api.cts.cdr.gov.au/cts/{conformanceId-guid}/dr/signin](https://api.cts.cdr.gov.au/cts/%7BconformanceId-guid%7D/dr/signin)  |
|  | **Sector\_Identifier\_Uri** | URL string referencing the client's sector identifier URI, used as an optional input to the Pairwise Identifier  | [https://api.cts.cdr.gov.au/{0}/cts/dr/sector1](https://api.cts.cdr.gov.au/%7B0%7D/cts/dr/sector1) |
|  | **GetDataRecipientStatus** | DH requests the data recipient status from the CTS Register via the Get Data Recipient Status Endpoint | [https://api.cts.cdr.gov.au/cts/{conformanceId}/register/cdr-register/v1/banking/data-recipients/status](https://api.cts.cdr.gov.au/cts/%7BconformanceId%7D/register/cdr-register/v1/banking/data-recipients/status) |
|  | **GetSoftwareProductStatus** | DH requests the software product status from the CTS Register via the Get Software Product Status Endpoint | [https://api.cts.cdr.gov.au/cts/{conformanceId}/register/cdr-register/v1/banking/data-recipients/brands/software-products/status](https://api.cts.cdr.gov.au/cts/%7BconformanceId%7D/register/cdr-register/v1/banking/data-recipients/brands/software-products/status) |
|  | **GetDataRecipients** | DH requests the data recipients from the CTS Register via the Getdatarecipients Endpoint | [https://api.cts.cdr.gov.au/cts/{conformanceId}/register/cdr-register/v1/banking/data-recipients](https://api.cts.cdr.gov.au/cts/%7BconformanceId%7D/register/cdr-register/v1/banking/data-recipients)  |
| Consent | **Authorise** | CTS ADR requests authorisation with the DH via the Authorise Endpoint | <https://www.dataholder.com.au/authorize>(Data Holder endpoint) |
|  | **Token** | CTS ADR exchanges their code for a Token from the DH via the Token Endpoint CTS ADR exchanges their Refresh Token for an Access Token from the DH via the Token Endpoint | <https://www.dataholder.com.au/token>(Data Holder endpoint) |
|  | **Introspection** | CTS ADR sends an Introspection request to the Data Holder Token Introspection endpoint to retrieve information about a token | <https://www.dataholder/token/introspection>(Data Holder endpoint) |
|  | **PAR** | DH receives a request with an Client Assertion via their PAR | <https://dataholder/par>(Data Holder endpoint) |
|  | *UserInfo***Not Built** | *CTS ADR sends a UserInfo request to the Data Holder Token UserInfo endpoint to retrieve information about a User* | [*https://www.dataholder.com.au/token/userinfo*](https://www.dataholder.com.au/token/userinfo)*(Data Holder endpoint)* |
|  | **Banking/** | [Phase 1](#_API’S)[Phase 2](#_API’S)  | [https://dataholder/banking/](https://dataholder/banking/accounts)(Data Holder endpoint) |
|  | **Common/** | [Phase 1](#_API’S)[Phase 2](#_API’S) | <https://dataholder/common> (Data Holder endpoint) |
| Revocation | **Arrangement Revocation - DH to DR** | DH sends a request, using their cdr arrangement id, to the CTS ADR to withdraw Arrangement Consent | [https://api.cts.cdr.gov.au/cts/{conformanceId-guid}/dr/arrangements/revoke](https://api.cts.cdr.gov.au/cts/%7BconformanceId-guid%7D/dr/arrangements/revoke) |
|  | **Arrangement Revocation - DR to DH** | CTS ADR makes an Arrangement Revocation request to the DH Revocation Endpoint (registered uri) | <https://data.holder.com.au/arrangements/revo>ke(Data Holder endpoint) |
|  | **Token Revocation - DR to DH** | CTS ADR makes a Token Revocation request to the DH Token Revocation Endpoint (registered uri) | <https://data.holder.com.au/revocation> (Data Holder endpoint) |

# Glossary

This section provides a list of CTS specific terms and their meanings.

|  |  |
| --- | --- |
| Term | Meaning |
| ADR  | Accredited data recipient.  |
| Authenticate / authentication | When a consumer verifies themselves with a DH. For more information see: <https://consumerdatastandardsaustralia.github.io/standards/#authentication-flows>.  |
| Authorise / authorisation | A consumer confirming to the disclosure of their CDR data from a DH. For more information see: <https://openid.net/specs/openid-connect-core-1_0.html#Overview>.  |
| Brand  | A DH’s system that is designed to interact with an ADR’s software product.   |
| CDR  | Consumer Data Right.  |
| CDS  | Consumer Data Standards.  |
| Consent | Used to refer to when a consumer agrees to share their CDR data with an ADR for a specific purpose (i.e. collect and use); technically distinguished from the final affirmative action (i.e. authorise) in the consent flow. Consent is also used as a term in consumer-facing interactions to refer to data sharing arrangements. Consent requirements will be communicated between the ADR and DH via the authorisation request object. The primary mechanism for capturing consent will be scopes and claims under Open ID connect. Other patterns for the establishment of consent may be considered in the future, including the incorporation of fine-grained consent for specific use cases. For more information see: <https://consumerdatastandardsaustralia.github.io/standards/#consent>.  |
| CTS  | Conformance Test Suite.  |
| CTS data recipient  | The ADR built within CTS. Used to test a DH’s brand during on-boarding. |
| CTS system | The components of the CTS which ADR and DH will interact with during conformance testing. |
| CTS Register | CDR Register functionality that has been replicated within CTS. Used for testing ADR software products and DH brands during on-boarding. |
| DH  | Data holder.  |
| E2E testing  | Refers to a software testing method that involves testing an application's workflow from beginning to end. |
| Revoke / revocation  | Revocation endpoint. DH and DR MUST implement an Arrangement Revocation endpoint as described in the Consumer Data Standards Endpoints. The Arrangement Revocation endpoint is used to revoke an existing sharing arrangement. DH MUST implement a token revocation end point as described in section 2 of [RFC7009]. The revocation end point serves as a revocation mechanism that allows an ADR to invalidate its tokens as required to allow for token clean up. Revocation of refresh tokens and access tokens MUST be supported. For more information see: * [https://consumerdatastandardsaustralia.github.io/standards/#end-points](https://consumerdatastandardsaustralia.github.io/standards/%23end-points)
* <https://tools.ietf.org/html/rfc7009#section-2>.
 |
| Software product | A software product developed by an ADR that is designed to interact with a DH’s brand to facilitate consent and request consumer data. |
| Test run   | A single instance of E2E testing that a participant will complete, resulting in a report for the CDR Register (the Register) to consider in allowing the participant to be active on the Register.  |
| Withdrawal | When a consumer stops a data sharing arrangement (i.e. consent/authorisation). This can occur via an ADR or a DH. This was previously referred to as **revocation**.  |

# Brand vs Conformance ID Infographic

The below diagram illustrates how each brand has its own Conformance Id / testing workflow.

1. If your brand checks the ADR register status by polling the register GetSoftwareProductStatus, scenario 11a will be assigned [↑](#footnote-ref-2)
2. If your brand checks the ADR register status by polling the register GetDataRecipients scenario 11b will be assigned [↑](#footnote-ref-3)