Data Standards Body
Technical Working Group

Decision Proposal 121 – Enhanced Error Handling – Application of HTTP Response Codes

Contact: Mark Verstege
Publish Date: 3rd June 2020
Feedback Conclusion Date:3rd July 2020

Context

The Data Standards define a subset of the allowable HTTP Status Codes and their applicability to the standards and CDR Register. Additional HTTP status codes have been identified under Decision Proposal 122 as part of a dependent consultation on enhanced error handling.

This proposal specifically relates to the use of HTTP Response Codes across the Data Standards and CDR Register.

This proposal includes the consideration of new HTTP status codes considered under Decision Proposal 122 - Extension of Supported HTTP Response Codes for Enhanced Error Handling.

It is considered in conjunction with a group of decision proposals under the Enhanced Error Handling problem space:

- Decision Proposal 119 - Enhanced Error Handling Payload Conventions
- Decision Proposal 120 - CDR Error Codes for Enhanced Error Handling
- **Decision Proposal 121 - Application of existing HTTP Error Response Codes to Enhanced Error Handling** (this proposal)
- Decision Proposal 122 - Extension of Supported HTTP Response Codes for Enhanced Error Handling
- Decision Proposal 127 - CX Guidelines for Enhanced Error Handling

Decision To Be Made

- Which HTTP response codes map to given error scenarios

Current recommendation

The recommendations of the Data Standards Body are supplied as a series of tables defining the HTTP status codes as applicable to the Data Standards and CDR Register, mapping the HTTP status codes to API endpoints and mapping high level error scenarios to HTTP status codes.

The DSB is seeking feedback on this recommendation. In particular, feedback on whether the recommendations provide sufficient detail for implementing complex error scenarios.

HTTP Response Codes

The following are the HTTP status codes for the different HTTP methods, across all Read/Write API endpoints. Additions to the existing data standards are included in green.

Situation	HTTP Status	Notes	POST	GET	DELETE
Query completed successfully	200 OK	Includes when a request is made to a Scheduled Payments or Direct Debits API, the Data Holder **must** respond with a 200 OK and an empty response.	**Yes**	**Yes**	No
Normal execution. The request has succeeded.	201 Created	The operation results in the creation of a new resource.	**Yes**	No	No
Delete operation completed successfully	204 No Content		No	No	**Yes**
The response is not modified since last call	304 Not Modified	May be returned if standard caching headers such as ETag or If-modified-since are utilised	**Yes**	**Yes**	No
Request has invalid, malformed, missing or non-compliant JSON body or URL parameters	400 Bad Request	The requested operation will not be carried out.	**Yes**	**Yes**	**Yes**
Authorization header missing or invalid token	401 Unauthorized	The operation was refused access. Re-authenticating may result in an appropriate token that may be used.	**Yes**	**Yes**	**Yes**
Token has incorrect scope or a security policy was violated.	403 Forbidden	The operation was refused access. Re-authenticating is unlikely to remediate the situation. It is expected that this error will result in an error payload	**Yes**	**Yes**	**Yes**
The requested URL does not exist, is not defined in the data standards or has not been implemented by the server. Equally, a requested resource identifier in the URL path does not exist or the server is not willing to disclose for business reasons.	404 Not Found	*Applies to the URL for POST and DELETE methods **only.** If a resource identifier is supplied in the request body and it does not exist or unable to be disclosed a 422 Unprocessable Entity applies.	Yes*	Yes	Yes*

The consumer tried to access the resource with a method that is not supported.	405 Method Not Allowed		**Yes**	**Yes**	**Yes**
The request contained an Accept header other than permitted media types, a character set other than UTF-8 or a version that was not supported	406 Not Acceptable		**Yes**	**Yes**	**Yes**
The operation was refused because the payload is in a format not supported by this method on the target resource.	415 Unsupported Media Type		**Yes**	No	No
The request was well formed but was unable to be processed due to business logic specific to the request	422 Unprocessable Entity	If applicable to the HTTP method it is expected that this error will result in an error payload	**Yes**	**Yes**	No
The operation was refused as too many requests have been made within a certain timeframe.	429 Too Many Requests	Throttling is an NFR. The data holder should include a Retry-After header in the response indicating how long the data consumer must wait before retrying the operation.	**Yes**	**Yes**	**Yes**
Something went wrong on the API gateway or micro-service	500 Internal Server Error	The operation failed.	**Yes**	**Yes**	**Yes**
Service is currently unavailable	503 Service Unavailable		**Yes**	**Yes**	**Yes**
The server was unable to respond in a timely manner	504 Gateway Timeout	Returned if a timeout has occurred but a resend of the original request is viable (otherwise use 500 instead)	**Yes**	**Yes**	**Yes**

High level error scenarios

Scenarios are listed in order of precedence however it is not a hard and fast rule. There are situations where the logic (e.g. rate limiting or signature validation) may be implemented at one of many layers in front of the API's application logic.

#	Area	Issue	Situation	HTTP Status Code
#1	**Authorisation**	Invalid Signature	The signature used to sign a private JWT or other data is invalid or not recognised	401 (Unauthorised)
#2	**Authorisation**	Security Condition Failed	A security condition prevents the request but the reason will not be exposed.	403 (Forbidden)
#3	**Data Request**	Not Found	The URL requested is not defined by the data standards or CDR Register, or it is not implemented by the server.	404 (Not Found)
#4	**Authorisation**	Invalid ADR	The ADR or ADR software product is invalid or not active in the CDR Register.	403 (Forbidden)
#5	**Authorisation**	Invalid DH	The DH is invalid or not active in the CDR Register.	403 (Forbidden)
#6	**Authorisation**	No Consent Established	Data is requested without valid consent, or an invalid (or missing) token is provided.	401 (Unauthorised)
#7	**Authorisation**	Invalid Token or Credential	The contents of the signed JWT, SSA or other client assertion is invalid or malformed.	401 (Unauthorised)
#8	**All**	Service Unavailable	The server is partially or wholly unavailable and the requested endpoint is currently not available.	503 (Service Unavailable)
#9	**Authorisation**	Consent Withdrawn	The ADR attempts to perform a request however the consent related to the token is withdrawn, expired or revoked	401 (Unauthorised)
#10	**Authorisation**	Invalid Consent	The ADR attempts to request data without the necessary scope or consent permissions	403 (Forbidden)
#11	**Data Request, Authorisation**	Too Many Requests	An ADR requests data or attempts to establish consent however one of the rate limiting NFR thresholds is reached.	429 (Too Many Requests)

#	Area	Issue	Situation	HTTP Status Code
#12	**Data Request, Authorisation**	Method Not Allowed	The URL is valid but the method is not supported (e.g. a PUT is requested for a GET-only endpoint)	405 (Method Not Allowed)
#13	**Data Request, Authorisation**	Requested Response Format Is Not Supported	The requested response format is not supported by the server (e.g. requested application/xml but the endpoint only supports application/json).	406 (Not Acceptable)
#14	**Data Request, Authorisation**	Requested Response Charset is Not Supported	The requested character set is not supported by the server.	406 (Not Acceptable)
#15	**Data Request, Authorisation**	Unsupported Media Type	The URL is valid but the payload is an unsupported format not accepted or not recognised by the server	415 (Unsupported Media Type)
#16	**Data Request**	Invalid Field Invalid Header Invalid Body Parameter	A field, header or request body parameter is malformed or an incorrect type.	400 (Bad Request)
#17	**Data Request**	Unsupported Version	A version is requested that is well-formed but unsupported by the server (i.e. positive integer but lower or higher than the supported version implemented by the server).	406 (Not Acceptable)
#18	**Data Request**	Unexpected Field Unexpected Header Unexpected Body Parameter	A field, header or request body parameter is provided but the endpoint excludes it or excludes variations to the interface contract.	400 (Bad Request)
#19	**Data Request**	Missing Field Missing Header Missing Body Parameter	A field, header or request body parameter is expected / mandatory however it is not included in the request.	400 (Bad Request)

#	Area	Issue	Situation	HTTP Status Code
#20	**Data Request**	Invalid Field Invalid Header Invalid Body Parameter	A field, header or request body parameter's value doesn't comply with the constraints of the field or is not an allowed value.	400 (Bad Request)
#21	**Data request**	Invalid Account	The account requested in the URL is invalid or the Data Holder is unwilling to disclose for business reasons.	404 (Not Found)
#22	**Data Request**	Invalid Industry	The industry requested in path to the CDR Register or Data Holder is invalid, does not exist and thus cannot be found.	404 (Not Found)
#23	**Data Request**	Not Found	The resource identifier requested in the URL does not exist or the server is unwilling to disclose it.	404 (Not Found)
#24	**Data request**	Invalid Account	The account requested in the request body is invalid, does not exist or the Data Holder is unwilling to disclose for business reasons.	422 (Unprocessable Entity)
#25	**Data Request**	Not Found	The resource identifier requested in the request body is invalid, does not exist or the Data Holder is unwilling to disclose for business reasons.	404 (Not Found)
#26	**Data request**	Closed Account	An ADR requests scheduled payment data or direct debits data for a closed account.	200 (OK), empty response
#27	**All**	Request Timeout	A request was made however the server was not able to respond in a timely manner.	504 (Gateway Timeout)
#28	**All**	Unexpected Error	Something went wrong at the server that was unexpected.	500 (Internal Server Error)

HTTP Response Code Mapping to End Points

The following are the HTTP status codes applicable for each API endpoint in the Data Standards and CDR Register **not covered** by a normative reference.

API Category	Endpoint	Applicable HTTP Response Codes
Banking	Get Accounts	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Banking	Get Bulk Balances	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Banking	Get Balances For Specific Accounts	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 405 (Method Not Allowed) • 422 (Unprocessable Entity) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx

API Category	Endpoint	Applicable HTTP Response Codes
Banking	Get Account Balance	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 404 (Not Found) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Banking	Get Account Detail	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 404 (Not Found) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 429 (Too Many Requests) • 5xx
Banking	Get Transactions For Account	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 404 (Not Found) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Banking	Get Transaction Detail	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 404 (Not Found) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx

API Category	Endpoint	Applicable HTTP Response Codes
Banking	Get Direct Debits For Account	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 404 (Not Found) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Banking	Get Bulk Direct Debits	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Banking	Get Direct Debits For Specific Accounts	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 405 (Method Not Allowed) • 422 (Unprocessable Entity) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Banking	Get Scheduled Payments for Account	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 404 (Not Found) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx

API Category	Endpoint	Applicable HTTP Response Codes
Banking	Get Scheduled Payments Bulk	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Banking	Get Scheduled Payments For Specific Accounts	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 405 (Method Not Allowed) • 422 (Unprocessable Entity) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Banking	Get Payees	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Banking	Get Payee Detail	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 404 (Not Found) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx

API Category	Endpoint	Applicable HTTP Response Codes
Banking	Get Products	• 200 (OK) • 400 (Bad Request) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Banking	Get Product Detail	• 200 (OK) • 400 (Bad Request) • 404 (Not Found) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Common	Get Customer	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Common	Get Customer Detail	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx

API Category	Endpoint	Applicable HTTP Response Codes
Common	Get Status	• 200 (OK) • 400 (Bad Request) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 418 (I'm A Teapot) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Common	Get Outages	• 200 (OK) • 400 (Bad Request) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Admin	Get Metrics	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Admin	Post Metadata Update	• 200 (OK) • 201 (Created) • 304 (Not Modified) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx

API Category	Endpoint	Applicable HTTP Response Codes
Consent	Delete CDR Arrangement	• 204 (No Content) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 404 (Not Found) • 405 (Method Not Allowed) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Register	Get Data Holder Brands	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 404 (Not Found) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Register	Get Software Statement Assertion	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 404 (Not Found) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Register	Get Software Product Status	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 404 (Not Found) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx

API Category	Endpoint	Applicable HTTP Response Codes
Register	Get Data Recipients Status	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 404 (Not Found) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx
Register	Get Data Recipients	• 200 (OK) • 400 (Bad Request) • 401 (Unauthorized) • 403 (Forbidden) • 404 (Not Found) • 405 (Method Not Allowed) • 406 (Not Acceptable) • 415 (Unsupported Media Type) • 422 (Unprocessable Entity) • 429 (Too Many Requests) • 5xx

Options Identified

These considerations depend on the recommendations made in Decision Proposal 122 - Extension of Supported HTTP Response Codes for Enhanced Error Handling.

Normative References

The handling of concerns such as oAuth token validation is handled by the normative references in the standards. The data standards do not seek to define CDR Error Codes where the normative standards apply. This ensures that normative references can be relied upon for interpretation and implementation based on the underlying standards themselves.

The counter position to this would make it hard to maintain the data standards because there would be highly coupled dependencies to the version of the normative references as well as the fact that it is unlikely to be practical to implement where organisations use off the shelf technologies to provide components of their solution such as Identity and Access Management or API Gateway software. In these cases error handling typically has limited flexibility to customise.

When to use 404 vs 501 vs 405

There are three situations where a requested resource does not exist, and it is worth informing the client of this situation:

1. The requested resource is not implemented. Perhaps the implementation compliance obligation is still in the future or the endpoint is optional to implement.
2. The requested resource is not part of the data standards. Perhaps the client is incorrectly requesting an endpoint but there is a typo in the URL or the API has been decommissioned and is no longer part of the data standards.
3. The requested resource does not exist. The URI path may be valid but a path parameter (e.g. resource Id) may not exist or perhaps it cannot be disclosed for business reasons. In this instance it has the same effect as the resource not being implemented.

To respond correctly, there are three HTTP status codes which may imply a resource has not been implemented in some fashion:

1. **404 Not Found** when the resource does not exist.
2. **405 Method Not Allowed** when the request method (e.g. GET, POST, etc.) is not allowed for the requested resource.
3. **501 Not Implemented** when the request method is not implemented by the resource server (e.g. the resource server cannot support the OPTIONS method).

501s should only be used when "the server does not support the functionality required to fulfil the request", e.g. the client is requesting a PATCH on an endpoint but the server does not support that operation at all.

A 501 status can also send a Retry-After header, telling the client when to check back to see if the functionality is supported by then.

501 is the appropriate response when the server does not recognise the request method and is incapable of supporting it for any resource. The only methods that servers are required to support (and therefore that must not return 501) are GET and HEAD.

If the server does recognise the method, but intentionally does not support it, the appropriate response is 405 Method Not Allowed.

On the other hand, 404s are designed for situations when the server does not offer the representation requested for a target resource. **404s should clearly be used when a given resource endpoint has not been implemented or does not exist (e.g. it is not part of the standards).**

6.6.2. 501 Not Implemented

> The 501 (Not Implemented) status code indicates that the server does not support the functionality required to fulfill the request. This is the appropriate response when the server does not recognize the request method and is not capable of supporting it for any resource.
>
> A 501 response is cacheable by default; i.e., unless otherwise indicated by the method definition or explicit cache controls (see Section 4.2.2 of [RFC7234]).

6.5.4. 404 Not Found

> The 404 (Not Found) status code indicates that the origin server did not find a current representation for the target resource or is not willing to disclose that one exists. A 404 status code does not indicate whether this lack of representation is temporary or permanent; the 410 (Gone) status code is preferred over 404 if the origin server knows, presumably through some configurable means, that the condition is likely to be permanent.
>
> A 404 response is cacheable by default; i.e., unless otherwise indicated by the method definition or explicit cache controls (see Section 4.2.2 of [RFC7234]).

6.5.5. 405 Method Not Allowed

> The 405 (Method Not Allowed) status code indicates that the method received in the request-line is known by the origin server but not supported by the target resource. The origin server MUST generate an Allow header field in a 405 response containing a list of the target resource's currently supported methods.
>
> A 405 response is cacheable by default; i.e., unless otherwise indicated by the method definition or explicit cache controls (see Section 4.2.2 of [RFC7234]).

Error Conditions

#	Situation	Example	Error Handling
1	**Resource is not valid**	GET data.holder.com.au/ cds-au/v1/banking/**foo/bar**	If the Client tries to access a URL for a resource that is not defined by the CDS specification, the Resource Server **should** choose to respond with a 404 (Not Found) and **must** be consistent with how they handle all other 404 (Not Found) conditions. It is noted that making this a MUST may not always be feasible where an API Gateway could be generically handling these issues.
2	**Resource has not been implemented**	DELETE data.recipient.com.au/ apis/cds-au/sharing-arrangement	If a Resource Server has not implemented an API endpoint, it **should** respond with a 404 (Not Found) for requests to that URL and **must** be consistent with how they handle all other 404 (Not Found) conditions. e.g. a Data Recipient only supports token revocation for consent management (revocation_endpoint) and has not yet implemented the CDR Arrangement API endpoint for concurrent consent. It is noted that making this a MUST may not always be feasible where an API Gateway could be generically handling these issues.

#	Situation	Example	Error Handling
3	**Resource URL does not exist or cannot be found**	GET data.holder.com.au/ banking/accounts/**19b8ec7809** GET data.holder.com.au/ banking/products/**ab829ef189**	When a Client tries to requests a resource URL with a resource Id that does not exist, the Resource Server **should** respond with a 404 (Not Found), rather than respond with a 422 (Unprocessable Entity) and **must** be consistent with how they handle all other 404 (Not Found) conditions. e.g. if the ADR tries to GET /banking/accounts/19b8ec7809 where **19b8ec7809** is not a valid accountId. Based on guidance provided in Issue #174 It is noted that making this a MUST may not always be feasible where an API Gateway could be generically handling these issues.

4	Resource in Request Body does not exist or cannot be found	POST data.holder.com.au/banking/
		accounts/balances
		
		{
		"data": {
		"accountIds": [
		"19b8ec7809"
		]
		},
		"meta": {}
		}
			When a Client tries to requests a resource within a request body but the requested resource does not exist, the Resource Server **must** respond with a 422 (Unprocessable Entity), rather than a 404 (Not Found).
			e.g. if the ADR tries to GET /banking/accounts/balances where **accountId=19b8ec7809** is provided in the request body and is not a valid accountId.

#	Situation	Example	Error Handling
5	**Resource in URL exists but business rules prevent sharing**	GET data.holder.com.au/banking/ accounts/**9fe8717ca89**	When a Client tries to request a resource URL with a resource Id that exists but business rules prevent sharing the data for some reason, the Resource Server **should** choose to respond with a 404 (Not Found), rather than respond with a 422 (Unprocessable Entity) and **must** be consistent with how they handle all other 404 (Not Found) conditions. e.g. if the ADR tries to GET /banking/accounts/0da594ec where **0da594ec** has an Account Frozen status or a fraud lock flag.

6	**Resource in Request Body exists but business rules prevent sharing**	POST data.holder.com.au/banking/
		accounts/direct-debits
		
		{
		"data": {
		"accountIds": [
		"9fe8717ca89"
		]
		},
		"meta": {}
		}
			When a Client tries to requests a resource within a request body which exists but business rules prevent sharing the data for some reason, the Resource Server **must** respond with a 422 (Unprocessable Entity), rather than a 404 (Not Found).
			e.g. if the ADR tries to GET /banking/accounts/direct-debits where **accountId=0da594ec** is provided in the request body has an Account Frozen status or a fraud lock flag.

#	Situation	Example	Error Handling
7	Resource in URL or Request Body exists but there is no business data associated with the resource	GET data.holder.com.au/banking/accounts/0da594ec/transactions/86ea2570-57af-4a86-917c-87b9b7d2be72	When a Client tries to request a resource URL that results in no business data being returned the Resource Server **must** respond with a 200 (OK) and set the data object to be empty. e.g. An ADR requests the transaction details for a transaction Id that exists but there is no detail for the transaction, GET /banking/accounts/{accountId}

When to use 422 vs 404 vs 400 vs 200

There are a variety of considerations that factor into the correct response:

- security considerations where a resource server may choose not to tell a client that the resource couldn't be found or be processed
- whether the resource is a protected resource (authenticated) or unprotected (unauthenticated)
- the resource exists but there are other conditions preventing the sharing of data
- the request was well formed but was unable to be processed due to business logic specific to the request
- the resource doesn't exists because it has not been implemented
- the resource isn't part of the data standards
- the method with which the resource is requested

In simple terms, the proposed change to the standards to is make clear where the request is a GET then 404 applies to the scenarios where the URI path is not found. This includes where a resource is requested using an identifier in the path (e.g Get Account Details).

Response Code handling

Method of request	**Resource**	**Description**	**Error Handling**
GET POST PUT DELETE	Path	A resource is requested in the URL path but it does not exist. Refer to error conditions.	If the Client tries to request a resource by URL that does not exist, the Resource Server **should** choose to respond with a 404 (Not Found).
POST PUT DELETE	Request Body	A resource is requested but it does not exist (refer to error conditions below) when it is supplied in the request body. Refer to error conditions.	A resource server must respond with one of the options below. In either option the resource server **must** provide an error list with a separate error item specifying each of the resources that do not exist or cannot be provided. When more than one resource Id is requested (e.g. Account ID) but some of the resource Ids are ok then the resource server **may** return a 200 (OK) with the response providing the list of resources that can be found and can be disclosed. The error object **must** be populated with the list of resource Ids that cannot be returned. When one or more resource Ids are requested the resource server **may** return a 422 (Unprocessable Entity). The error object **must** be populated with the list of resource Ids that cannot be returned.

Error Conditions

#	Situation	Example	Error Handling
1	**Resource is not valid**	GET data.holder.com.au/cds-au/v1/ banking/**foo/bar**	If the Client tries to access a URL for a resource that is not defined by the CDS specification, the Resource Server **should** choose to respond with a 404 (Not Found) and **must** be consistent with how they handle all other 404 (Not Found) conditions. It is noted that making this a MUST may not always be feasible where an API Gateway could be generically handling these issues.
2	**Resource has not been implemented**	DELETE **data.recipient.com.au/** **apis/cds-au/v1/** **sharing-arrangement**	If a Resource Server has not implemented an API endpoint, it **should** respond with a 404 (Not Found) for requests to that URL and **must** be consistent with how they handle all other 404 (Not Found) conditions. e.g. a Data Recipient only supports token revocation for consent management (revocation_endpoint) and has not yet implemented the CDR Arrangement API endpoint for concurrent consent. It is noted that making this a MUST may not always be feasible where an API Gateway could be generically handling these issues.
3	**Resource URL does not exist or cannot be found**	GET data.holder.com.au/cds-au/v1/ banking/accounts/**19b8ec7809** GET data.holder.com.au/cds-au/v1/ banking/products/**ab829ef189**	When a Client tries to requests a resource URL with a resource Id that does not exist, the Resource Server **should** respond with a 404 (Not Found), rather than respond with a 422 (Unprocessable Entity) and **must** be consistent with how they handle all other 404 (Not Found) conditions. e.g. if the ADR tries to GET /banking/accounts/19b8ec7809 where **19b8ec7809** is not a valid accountId. Based on guidance provided in Issue #174 It is noted that making this a MUST may not always be feasible where an API Gateway could be generically handling these issues.

#	Situation	Example	Error Handling

4	Resource in Request Body does not exist or cannot be found	POST data.holder.com.au/cds-au/v1/
		banking/accounts/balances
		{
		"data": {
		"accountIds": [
		"19b8ec7809"
		]
		},
		"meta": {}
		}
			When a Client tries to requests a resource within a request body but the requested resource does not exist, the Resource Server **must** respond with a 422 (Unprocessable Entity), rather than a 404 (Not Found).
			e.g. if the ADR tries to GET /banking/accounts/balances where **accountId=19b8ec7809** is provided in the request body and is not a valid accountId.

5	**Resource in URL exists but business rules prevent sharing**	GET data.holder.com.au/cds-au/v1/ banking/accounts/**9fe8717ca89**	When a Client tries to request a resource URL with a resource Id that exists but business rules prevent sharing the data for some reason, the Resource Server **should** choose to respond with a 404 (Not Found), rather than respond with a 422 (Unprocessable Entity) and **must** be consistent with how they handle all other 404 (Not Found) conditions. e.g. if the ADR tries to GET /banking/accounts/0da594ec where **0da594ec** has an Account Frozen status or a fraud lock flag.

6	**Resource in Request Body exists but business rules prevent sharing**	POST data.holder.com.au/cds-au/v1/
		banking/accounts/direct-debits
		{
		"data": {
			When a Client tries to request a resource within a request body which exists but business rules prevent sharing the data for some reason, the Resource Server **must** respond with a 422 (Unprocessable Entity), rather than a 404 (Not Found).

#	Situation	Example	Error Handling
		"accountIds": ["9fe8717ca89"] }, "meta": {} }	e.g. if the ADR tries to GET /banking/accounts/direct-debits where **accountId=0da594ec** is provided in the request body has an Account Frozen status or a fraud lock flag.
7	Resource in URL or Request Body exists but there is no business data associated with the resource	GET data.holder.com.au/cds-au/v1/banking/accounts/**0da594ec/transactions/86ea2570-57af-4a86-917c-87b9b7d2be72**	When a Client tries to request a resource URL that results in no business data being returned the Resource Server **must** respond with a 200 (OK) and set the data object to be empty. e.g. An ADR requests the transaction details for a transaction Id that exists but there is no detail for the transaction, GET /banking/accounts/{accountId}

When to use 403 vs 404 or 422

Background

Typically, developers want to provide clear and deterministic error codes that allow for procedural handling of each error individually. This aids clear reporting of issues to the user, gracefully degradation of the UX and providing a better quality of service.

Sometimes however, providing too much detail can result in unintended security side effects. One such example is increasing API fuzzing risk. In such a scenario an attacker can farm valid resource IDs by hitting a GET endpoint with a valid authentication session.

Example: let's apply this example to a Get Balances For Specific Accounts.

GET data.holder.com.au/cds-au/v1/banking/accounts/balances

Customer	Account ID
Bob	111
Jane	222

Bob can authenticate and hit the GET Get Balances For Specific Accounts endpoint with incremental account ids. Suppose descriptive HTTP status codes were used as follows:

- If the Account ID exists then the GET returns a 200.
- If the Account ID does not exist then the GET returns a 404.
- If the Account ID exists but a business rule prevents sharing, the GET returns a 422.
- If the Account ID exists but it is not owned by Bob then the GET returns 403.

This means that Bob can brute force the endpoint and farm a list of valid Account IDs.

Risk of Exposing API Resource Identifiers

Probing Risk

An attacker can farm Ids and then probe other endpoints with the known Ids. APIs should be correctly implementing their entitlements checks so the attacker should not be able to do anything with the Ids. If there is another bug, vulnerability or poor implementation that allows the Id to be used, there is the risk of combination attack.

Fuzzing Risk

An attacker tries to inject invalid, unexpected, or random data as inputs to see how the API responds. The attacker may identify different situations that result in different responses which leaks some security context of the application.

Combination attacks

An attacker combines one or more vulnerabilities to elevate the attack and gain access to more sensitive information. A probing attack may identify a valid account Id owned by Jane, a bank customer, who is not Bob. This attack is used in conjunction with a known bug in consent validation that does not check that the account being requested is associated with Bob's consent. The attacker can gain access to Jane's bank account details.

Error Conditions

#	Situation	Error Handling
1	**Protected resource is in the URL but the resource Id is not associated with the consumer's consent**	When a Client tries to request a protected (authenticated) resource URL with a resource Id that is not associated with the consumer's consent contract for the given cdr_arrangement_id and access token, the Resource Server • **should** respond with a 404 (Not Found), • **must** not respond with a 403 (Forbidden) and • **must** be consistent with how they handle all other 404 (Not Found) conditions. This also covers the situation where the accountId represents another customer's account not associated with the primary subject's consent.
2	**Protected resource in the Request Body but the resource Id is not associated with the consumer's consent**	When a Client tries to request a protected (authenticated) resource where the resource Id is provided in the request body and the resource Id is not associated with the consumer's consent contract for the given cdr_arrangement_id and access token, the Resource Server • **must** respond with a (Unprocessable Entity), • **must** not respond with a 403 (Forbidden) This also covers the situation where the accountId represents another customer's account not associated with the primary subject's consent.

Threat Vectors and Security Conditions

Threats scenarios

Similar to the concerns raised in 403 vs 404/422 there are other security scenarios where a bank does not wish to disclose that an error, risk threshold or security condition has occurred. In these instances, providing a generic error response that does not leak security context is important. In other words, the situation shouldn't look atypical to the Client. An example may be a firewall rule identifies suspicious traffic, an adaptive authentication engine returns a risk score greater than a tolerable threshold. Examples of threat vectors include:

- IP Address is suspicious or blacklisted
- Geo-location is suspicious or blacklisted
- Security Risk Engine threshold exceeded or request marked as suspicious
- Any edge-security threat flag is raised

ADR status scenarios

An ADR and its associated software products must be active to facilitate a successful data sharing request. In the event that the ADR or one of its software products changes status, the Data Holder should not allow a data sharing request.

Error Conditions

#	Situation	Is the Access Token Valid?	Is Consent Active?	Is the Client Status OK?	Error Handling
1	Consent is active and the request to a protected resource is made with an invalid access token	**NO**	**YES**	**YES**	When a Client tries to request an authenticated resource URL with a resource Id that exists but the access token is not valid (e.g. expired), the Resource Server **must** respond with a 401 (Unauthorized) rather than a 403 (Forbidden). e.g. the lifetime of the access token has been exceeded
2	**Consent is not active and the request to a protected resource is made with a valid access token**	**YES**	**NO**	**YES**	When a Client tries to request an authenticated resource URL with a resource Id that exists but the access token is not valid (e.g. expired), the Resource Server **must** respond with a 403 (Forbidden) rather than a 401 (Unauthorized). e.g. a consumer has revoked consent in the Data Holder dashboard however an ADR has recently obtained an access token from the Data Holder and attempts to make a request.
3	The participant status is not active	**YES**	**YES**	**NO**	When the ADR or the ADR Software Product status is not active and the ADR tries to request an authenticated URL the Data Holder **must** respond with a 403 (Forbidden); When the Data Holder status is not active and the Data Holder tries to call an ADR URL ADR **must** respond with a 403 (Forbidden) https://cdr-register.github.io/register/#participant-statuses e.g. GET data.holder.com.au/cds-au/v1/banking/0da594ec DELETE data.recipient.com.au/apis/cds-au/sharing-arrangement/faa68f27-565d-49a6-a7af-5fac80a0e7d6

#	Situation	Is the Access Token Valid?	Is Consent Active?	Is the Client Status OK?	Error Handling
4	**The** resource is request but a security condition prevents the data request	?	?	?	The Resource Server should respond with an error that would be appropriate for the request being made. This situation is different to the resource itself being invalid or prevented from disclosure because of business rules. It is the condition where a security condition occurs prior to the resource's entitlements being checked. In order or precedence: If the access token is invalid, the Resource Server **must** respond with a 401 (Unauthorized) If the consent is not active, the Resource Server **must** respond with a 403 (Forbidden) If the Client participant status is not active, the Resource Server **must** respond with a 403 (Forbidden) The Client tries to request a resource URL but a security condition fails to be met on the Resource Server, The Resource Server **must** respond with a 403 (Forbidden) or 404 (Not Found). The Client tries to request a resource supplying the resource identifier(s) in the request body but a security condition fails to be met on the Resource Server, The Resource Server **must** choose to respond with a 403 (Forbidden) or 422 (Unprocessable Entity).

When to use 406 vs 400

406 is used in situations where a version is requested but is not supported by the resource server. In other situations the standards specify the use of 400 (Bad Request) where a field is invalid or malformed. 406 should only be used for negotiation based on header fields. Currently this only covers the version headers (x-v, x-min-v) and Accept header. The remaining headers are not used to negotiate a specific representation of the request so they are covered by 400 (Bad Request) or the relevant HTTP status code based on the error encountered.

> **6.5.6. 406 Not Acceptable**
>
> The 406 (Not Acceptable) status code indicates that the target resource does not have a current representation that would be acceptable to the user agent, according to the proactive negotiation header fields received in the request (Section 5.3), and the server is unwilling to supply a default representation.

Consent Revocation

In the CDR Rules, Data Holders must notify Data Recipients when consent is withdrawn.

> **4.25 Withdrawal of authorisation to disclose CDR data and notification**
>
> (1) The CDR consumer who gave, to a data holder, an authorisation to disclose particular CDR data to an accredited person may withdraw the authorisation at any time:
> (a) by communicating the withdrawal to the data holder in writing; or
> (b) by using the data holder's consumer dashboard.
>
> (2) The data holder must:
> (a) if the withdrawal was in accordance with paragraph (1)(a)—give effect to the withdrawal as soon as practicable, and in any case within 2 business days after receiving the communication; and
> (b) in any case—notify the accredited person of the withdrawal in accordance with the data standards.

This process is facilitated through the CDR Arrangement API hosted by the Data Recipient. When a Data Holder encounters an error - say the Data Recipient is down for system maintenance - the consent is now revoked on the Data Holder side but not on the Data Recipient side. This creates a CX gap - the consumer will see consent as revoked on the Data Holder side, but they will see consent as active on the Data Recipient side. It should be noted that there is no risk to unauthorised disclosure of data from this point onwards because the Data Holder has marked consent as revoked. In this situation, a Data Holder would be expected to try communicating the consent revocation with the Data Recipient again within reasonable limits.

The corollary situation (Data Recipient revoking consent with the Data Holder) is slightly different. An ADR can clearly communicate that the consumer must try again (either immediately or at a later time). The reason that a Data Holder would want to cease data sharing as soon as a consumer notifies them is to prevent the ongoing disclosure of data when the consumer clearly does not wish this to continue. It may be through the Data Holder dashboard or it may be the receipt of communications from the consumer (e.g. a call to the data holder's contact centre).

To improve handling of this situation, informing Data Recipients in a meaningful way that consent is not active is important - e.g. when they attempt to collect data again without realising their consent is revoked.

This must consider three situations:

1. The ADR has a valid access token and tries to request consumer data.
2. The ADR doesn't have an access token and the ADR tries to request an access token from the DH's authorisation server using a refresh token.
3. The ADR doesn't have an access token or refresh token, they only have an authorisation code and the ADR tries to request a refresh token and access token from the DH's authorisation server using an authorisation code.

Error Conditions

#	Situation	Is the Access Token Valid?	Is Consent Active?	Error Handling
1	ADR tries to request a protected resource	**YES**	**NO**	When a Client tries to request an authenticated resource URL but consent is revoked, the Resource Server **must** respond with a 403 (Forbidden). The error response is covered by a CDR Error Code.
2	ADR tries to request an access token using the oAuth Token endpoint	N/A	**NO**	Error handling is governed by the normative standards [RFC6749 oAuth 2.0]. An **invalid_grant** grant error must be returned.

3	ADR tries to obtain tokens using the oAuth Token endpoint with an authorisation code	N/A	**NO**	Error handling is governed by the normative standards [RFC6749 oAuth 2.0].
				An **invalid_grant** grant error must be returned.
				For example:
				`HTTP/1.1 400 Bad Request`
				`Content-Type: application/json;charset=UTF-8`
				`Cache-Control: no-store`
				`Pragma: no-cache`
				
				`{`
				`"error":"invalid_request"`
				`}`

Customer Records, Accounts and Consent

Accounts are associated to consent. Because of a variety of reasons, the accounts that a consumer elects to share with an ADR may change within a CDR sharing arrangement or may be unavailable for data sharing. Reasons may include:

- An account is a joint account and the second joint account holder has withdrawn their consent election
- An account is frozen or suspended by a bank because of suspicious payments activity
- A customer hasn't provided sufficient KYC evidence, or their KYC documents are old and have not been refreshed
- A bank prevents the disclosure of account data when their are customer vulnerability flags related to the account
- The consumer is locked out of internet banking
- A consumer removes accounts from their data sharing arrangement via the Data Holder Dashboard
- Other security reasons that prevent a bank from disclosing the data

In a similar context, the customer details requested via the Get Customer and Get Customer Details endpoints may be denied where business rules prevent disclosure.

Where the requested account(s) are not associated to consent or cannot be shared for business or security reasons, a Data Holder should respond with an error. Depending on the situation, this may be:

1. **A 404 (Not Found)** when the resource is requested via the URI path parameter specifying the account that could not be shared in the Response Error List
2. **A 422 (Unprocessable Entity)** when the resource(s) are supplied in the request body specifying the account(s) that could not be shared in the Response Error List.
3. **A 200 (OK)** with a partial or empty result set when more than one account Id is requested (e.g. via bulk GET). The Data Holder should populate the Response Error List errors[] array with the list of accounts that will not be shared along with the partial result set.

Response Error List requirements

When returning the errors encountered, a separate error item should be returned for each account or resource Id that cannot be returned.

```
## Valid response
{
  "errors": [
    {
      "code": "AU.CDR.Entitlements.InvalidAccount",
      "title": "Invalid Account",
      "detail": "29202ah34e"
    }, {
      "code": "AU.CDR.Entitlements.InvalidAccount",
      "title": "Invalid Account",
      "detail": "00284ae747",
      "meta": {
        ...
      }
    }
```

```
  ]
}
```

```
## Invalid response
{
  "errors": [
    {
      "code": "AU.CDR.Entitlements.InvalidAccount",
      "title": "Invalid Account",
      "detail": "Accounts 29202ah34e and 00284ae747 cannot be returned.",
      "meta": {
        ...
      }
    }
  ]
}
```

```
## Invalid response
{
  "errors": [
    {
      "code": "AU.CDR.Entitlements.InvalidAccount",
      "title": "Invalid Account",
      "detail": "29202ah34e, 00284ae747",
      "meta": {
        ...
      }
    }
  ]
}
```

Auditing

In all of these situations it is expected that a Data Holder would record the reasons and the rejection must be included in the Data Holder's metrics obligations.

Error Conditions

Under these sorts of circumstances, the expectations are presented in the following table:

#	Situation	Error Handling
1	The consumer removes election of the account in the Data Holder Dashboard	A meaningful reason for not sharing the account data should be provided in the error description. The account Id (as in the ID Permanence version) must be specified in the error description.
2	A Joint Account holder has withdrawn consent election to an account	Where an account cannot be shared because a joint account holder has withdrawn their consent election, a meaningful reason for not sharing the account data is provided in the error description. The account Id must be specified in the error description.
3	The consumer no longer owns the account	Where an account is no longer owned by the consumer, it must not be shared. If account ownership it is not considered a sensitive condition, a meaningful reason for not sharing the account data is provided in the error description. The account Id must be specified in the error description.
4	The account Id is invalid / not associated with the consumer's consent (including the **resource requested is just wrong)**	A meaningful reason for not sharing the account data is provided in the error description. The account Id must be specified in the error description.
5	A security exception or event prevents the Data Holder sharing an account	No reason must be disclosed. A generic error description must be provided. Data Holders must not share or disclose what events, triggers or rules are considered security exceptions. The account Id must be specified in the error description. A data holder may also respond with a 403 (Forbidden). No detail should be disclosed.

#	Situation	Error Handling
6	Sensitive business rules prevent the Data Holder sharing an account	No reason must be disclosed. A generic error description must be provided. Data Holders must not share or disclose what conditions or business rules are considered sensitive. The account Id must be specified in the error description.
7	Non-sensitive business rules prevent the Data Holder sharing an account	A meaningful reason for not sharing the account data is provided in the error description. The account Id must be specified in the error description.

Appendix: Issues considered

- https://github.com/ConsumerDataStandardsAustralia/standards/issues/11
- https://github.com/ConsumerDataStandardsAustralia/standards/issues/68
- https://github.com/ConsumerDataStandardsAustralia/standards/issues/119
- https://github.com/ConsumerDataStandardsAustralia/standards/issues/120
- https://github.com/ConsumerDataStandardsAustralia/standards/issues/121
- https://github.com/ConsumerDataStandardsAustralia/standards/issues/122
- https://github.com/ConsumerDataStandardsAustralia/standards-maintenance/issues/5
- https://github.com/ConsumerDataStandardsAustralia/standards-maintenance/issues/36
- https://github.com/ConsumerDataStandardsAustralia/standards-maintenance/issues/78
- https://github.com/ConsumerDataStandardsAustralia/standards-maintenance/issues/117
- https://github.com/ConsumerDataStandardsAustralia/standards-maintenance/issues/118
- https://github.com/ConsumerDataStandardsAustralia/standards-maintenance/issues/133
- https://github.com/ConsumerDataStandardsAustralia/standards-maintenance/issues/141
- https://github.com/ConsumerDataStandardsAustralia/standards-maintenance/issues/164
- https://github.com/ConsumerDataStandardsAustralia/standards-maintenance/issues/174
- https://github.com/ConsumerDataStandardsAustralia/standards-maintenance/issues/179
- https://github.com/ConsumerDataStandardsAustralia/standards-maintenance/issues/188