Data Standards Body
Technical Working Group

Decision Proposal 120 – Enhanced Error Handling – Error Code Catalogue

Contact: Mark Verstege
Publish Date: 12th June 2020
Feedback Conclusion Date: 10th July 2020

Context

Well-defined error handling is key to integration of ADRs and Data Holders. As the CDR ecosystem grows — soon to accommodate all Accredited Deposit-taking Institutions — the complexity for ADRs without well-defined and deterministic error handling grows. In turn, this will lead to inconsistent handling of errors by ADRs and a poorer consumer experience that may become confusing and lower trust in the CDR.

To provide predictable and scalable error handling across the ecosystem, Ensuring that there is clear standards to apply application logic for error handling in a consistent fashion is critical.

This proposal specifically relates to the definition of a catalogue of CDR error codes that apply to the Data Standards and CDR Register. 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** (this proposal)
- Decision Proposal 121 - Application of existing HTTP Error Response Codes to Enhanced Error Handling
- Decision Proposal 122 - Extension of Supported HTTP Response Codes for Enhanced Error Handling
- Decision Proposal 127 - CX Guidelines for Enhanced Error Handling

Background

Whilst HTTP status codes describe the nature of an error at a coarse-grained level, they provide little detail to the client about the specific error encountered. Because many unique error scenarios may be encountered, defining a common error catalogue that all participants must implement can improve consistency and reduce the complexity cost for API clients, especially Data Recipients.

HTTP status codes, whilst sufficient to describe the nature of an error at a coarse-grained level, are often times insufficient to convey enough information about an error to be helpful. Especially in an ecosystem such as the CDR where there are many server and many client implementations, being more specific when describing a domain-specific error can reduce complexity and inconsistency. Conveying the *application logic* of the CDR ecosystem is important to enable API clients to build against the business rules of the CDR.

Defining a CDR-specific catalogue of error codes provides predictability and specificity where commonality in error handling aides clients and consumers. This enables API clients to be informed of both the high-level error class (using the status code) and the finer-grained details of the problem (using one of the CDR error codes).

For example, consider a request to get the transaction for a consumer's bank account but that account has been removed from the consent arrangement and is no longer accessible The 422 Unprocessable Entity status code might inform the API client that the request encountered an error but it is generic and does not convey any business logic that resulted in the error. Thus, defining a catalogue of CDR Error codes is important to convey enough business logic to the API client to allow them to programmatically interpret the error and handle it gracefully. Equally, it ensures that API clients can rely on a consistent implementation across the many server implementations.

Whilst this is primarily to aide Data Recipient to Data Holder communications, there are other flows such as Data Recipient to CDR Register, Data Holder to Data Recipient and CDR Register to Data Holder communications that also benefit from implementation consistency.

To-date the data standards have defined a single error code relevant to data holder implementations. The CDR Register has not defined any error codes which may be returned from the CDR Register as part of servicing data recipient and data holder requests.

The standards have avoided defining common error codes until clear and explicit error scenarios are identified by the community. As the CDR moves closer to go-live for consumer data sharing, there are now participants with working implementations working on real-world integration and interoperability challenges. This has led to a number of defined error scenarios to be identified.

The key problem to solve is one based on need to define a set of common error codes that participants must adopt. The reasons the definition of this catalogue is important now is:

- it provides uniformity and consistency on error scenarios expected to be commonplace to all participants
- reduces the risk of implementation fragmentation moving forwards
- provides greater predictability for ADRs and better CX outcomes for graceful handling of errors to assist consumers in their user journey
- is defined based on a body of evidence from real-world implementation of the first stable version of the CDS and the lessons learnt
- ensures that the CDR ecosystem as it evolves to take on a large number of non-major ADIs can scale and continue to interoperate in a predictable and repeatable manner
- provides consumer experience guidance when errors are encountered and how they must be represented to consumers as well as the options for handling for the errors

Decision To Be Made

- Define the CDR Error Code Catalogue applicable to all software in the CDR ecosystem.

Requirements

1. There must be uniformity across the CDR Register and Data Standards
2. There must be flexibility to enable participants to extend the list of CDR error codes to their purposes
3. Participants must use CDR error codes where they are identified instead of creating a custom error code that represents the same error scenario
4. Error handling must consider security of the Data Holder and CDR regime
5. Error handling must consider the consumer's privacy and circumstances
6. Errors must be meaningful to API clients and consumers
7. Error codes must be defined in a way to be supportable across a large ecosystem of participants

Options Identified

Normative References

Option 1 – Don't change 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.

Option 2 – Override normative references to use CDR error codes and format

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.

Error Code Enumerations Catalogue

Option 1 – No change

Define no new error codes and maintain only the one error code currently defined by the data standards but map this to the agreed error code enumeration namespace changes.

Option 2 – Extend error catalogue to previously

Extend the error catalogue based on the error scenarios currently provided by the community. These are provided below in the Error Catalogue table.

Option 3 – Extend error catalogue to all CDR data flows

This extends option 2 but continues the work with the community to map out any other error codes required to cover known error scenarios that are not covered by normative references. This includes identifying error codes across:

- consent establishment,
- consent withdrawal and revocation,
- data sharing requests for once-off, ongoing and concurrent consent,
- CDR Register onboarding and CDR Register interactions,
- notification events between Data Recipients and Data Holders,
- NFRs and metrics reporting

Error Catalogue

MUST vs SHOULD vs MAY

The data standards follow the definitions in RFC2119 which provide the definition and use of key words defining requirements levels. It is expected that despite more specific guidance, some errors would not be MUSTs because the infrastructure of a given organisation may prevent the customisation of error codes. A good example of this is modern WAFs and API Gateways typically handle general errors, so they are handled consistently and reliably across an API catalogue. Errors like rate limiting, URIs not found because they haven't been published in the organisation's API Portal and the like are common scenarios where forcing a CDR Error Code may be unviable.

Reasonable effort should be made by all participants to fully implement the catalogue of error codes however there is necessary flexibility for participants. In saying that, there are error codes that are described as MUSTs because it is viable to cater for error handling logic within the application layer for the API.

Error Categories

Errors have been categorised into a collection of error categories that represent a set of similar issues. Generally, they will be handled in a similar fashion although they may result in different HTTP Status Codes and unique CDR Error Codes.

Error Category	Description
Register	This error category is reserved for use by the CDR Register only. Data Recipients and Data Holders **must not** use this error category for their custom error codes.
Invalid	The API request contains a structurally invalid parameter, field or header or it does not conform to the data type (either primitive or type definition) specified for the field (e.g. integer vs Base64 vs AmountString).

Error Category	Description
Constraint Violation	The API request contains a field-value that violates a constraint or condition for allowable values for that given field. For example, • "page=6" is requested but only 5 pages of transactions exist • A "product-category" filter request by a value that is not defined • The value provided does not conform to the regex for a field Currently error codes presented below are covered by the "Invalid" category however there may be merit in differentiating field-value errors differently to invalid field type or malformed fields.
Missing	An expected field, header or parameter was missing but it is required. This occurs when a field is mandatory or conditionally required and there is no default value that can be applied. The Resource Server cannot reliably service the request without the missing field because it would break the interface contract and/or cannot be reasonably interpreted.
Unexpected Field or Header	The API request was not expecting to receive a field, but the client made a request that included an unexpected field.
Data Validation	Whilst the request was well formed, the value is known to violate the allowable values or a source data constraint. Examples include the filtering of products where the product-category is not supported by the ADI.
Not Found	The API request is for an API that is currently unsupported by the Resource Server or not specified in the CDS.
Resource Entitlements	An entitlement of the consumer for a protected resource is not met. Entitlements conditions include: • The status of the ADR may not be valid, • Consent may be in a status that does not permit the successful execution of the request, • An account-level permission or entitlement prevents execution (e.g. a joint account holder has revoked consent election), • The resource is protected and cannot be found, • A Resource Server business rule prevents execution, • The Resource Server does not want to disclose if the resource exists • The customer no longer owns an account or is no longer a customer of the bank
Unsupported	The request is unsupported because of a qualification the client requested.
Unexpected Error	Something happened when processing the request but it was not expected.

Error Category	Description
Service Unavailable	The CDR service is down for system maintenance / scheduled outage or there is either a full or partial unscheduled outage. ADRs may check the Data Holder's Get Outages and Get Status endpoints for further details.

Error Catalogue

The error catalogue represents the suggested catalogue of errors based on the known error scenarios. Other error codes may be identified through further consultation.

Element	Definition
HTTP Status Code	The HTTP Status Code that maps to the CDR Error Code for the given error scenario
Error Code	Maps to the standard error code structure: /errors/error[x]/code
Error Code Title	Maps to the standard error code structure: /errors/error[x]/title
Error Scenario	The specific error scenario raised by the community and how it must be handled

#	HTTP Status Code	Error Code	Error Code Title	Error Scenario
1.	400 (Bad Request)	AU.CDR.Missing.Field	Missing Required Field	The request is missing a mandatory field required for the API. It may be a missing query parameter or missing field in the request payload. This error code can be used, if it is not already captured under a specialising validation. Reference of the missing field should be provided in the error description explaining the specific error encountered. Applies to: • Query parameters • Request body parameters
2.	400 (Bad Request)	AU.CDR.Missing.Header	Missing Required Header	A required HTTP header has not been provided. The HTTP Header should be specified in the error description. Applies to: • Header parameters
3.	400 (Bad Request)	AU.CDR.Unexpected.Field	Unexpected Field Not Allowed	When a query parameter or request body field-value is provided but the API forbids the use of that field-value or any others but the ones specified by its interface contract. Reference of the unexpected field should be provided in the error description. Applies to: • Query parameters • Request body parameters

#	HTTP Status Code	Error Code	Error Code Title	Error Scenario
4.	400 (Bad Request)	AU.CDR.Unexpected.Header	Unexpected Header Not Allowed	When a HTTP Header is provided but the API forbids the use of that field-value or any others but the ones specified by its interface contract. The HTTP Header should be specified in the error description. Applies to: • Header parameters
5.	400 (Bad Request)	AU.CDR.Invalid.Field	Invalid Field	When the value of the URL field or request body field is an invalid type or the value violates the field's constraints as defined by the interface contract. For example, page-size is a PositiveInteger but a DateString is provided. Reference of the field should be provided in the error description along with the detail explaining the valid format. Applies to: • Query parameters • Path parameters • Request body parameters
6.	400 (Bad Request)	AU.CDR.Invalid.Header	Invalid Header	When a HTTP Header is provided but the field-value is an invalid type or violates the constraints as defined by the data standards. The HTTP Header should be specified in the error description. Applies to: • Header parameters

#	HTTP Status Code	Error Code	Error Code Title	Error Scenario
7.	400 (Bad Request)	AU.CDR.Invalid.DateTime	Invalid Date	An invalid date is supplied e.g. when a future date is expected, a date in past or current date is supplied. The message can specify the actual problem with the date. Reference of the invalid field should be provided in the description field explaining the valid behaviour. Applies to: • "oldest-time" query parameter • "newest-time" query parameter
8.	400 (Bad Request)	AU.CDR.Invalid.PageSize	Invalid Page Size	Page Number/Page size not a positive Integer Applies to: • "page-size" query parameter
9.	400 (Bad Request)	AU.CDR.Register.InvalidBrand	Invalid Brand	The brand provided to get the Data Recipient software statement assertion is invalid. Applies to: • dataRecipientBrandId path parameter for CDR Register APIs
10.	400 (Bad Request)	AU.CDR.Register.InvalidIndustry	Invalid Industry Requested	The industry requested in the path to get Data Recipient or Data Holder metadata is invalid / does not exist and cannot be found. Applies to: • "industry" path parameter for CDR Register APIs

#	HTTP Status Code	Error Code	Error Code Title	Error Scenario
11.	401 (Unauthorised)	AU.CDR.Entitlements.InvalidAdrStatus	ADR Status Is Invalid	The ADR or the ADR software product is not in an "active" state in the CDR Register Reference of the invalid ADR status should be provided in the description. Applies to: • All authenticated APIs
12.	401 (Unauthorised)	AU.CDR.Entitlements.InvalidAdrSoftwareProductStatus	ADR Software Product Status Is Invalid	The ADR or the ADR software product is not in an "active" state in the CDR Register. Reference of the invalid ADR Software Product status should be provided in the description. Applies to: • All authenticated APIs
13.	403 (Forbidden)	AU.CDR.Entitlements.InvalidConsentStatus	Consent Is Invalid	The resource's associated consent is not in a status that would allow the resource to be to be executed. E.g., if consent had been revoked but the data recipient tries to request the customer details. Reference of the consent should be provided in the description field explaining the reason consent is invalid. Applies to: • All authenticated APIs

#	HTTP Status Code	Error Code	Error Code Title	Error Scenario
14.	403 (Forbidden)	AU.CDR.Entitlements.ConsentIsRevoked	Consent Is Revoked	The consumer's consent is revoked, and the data request will not be serviced. It is a specialisation of AU.CDR.Entitlements.InvalidConsentStatus. Reference to the consent status should be provided in the description field, without revealing sensitive information. Applies to: • All authenticated APIs
15.	403 (Forbidden)	AU.CDR.Entitlements.Forbidden	Resource Is Forbidden	A business or security condition prevents the request and it has been forbidden. Reference to the specific error encountered should be provided in the description field, without revealing sensitive information. Applies to: • All authenticated APIs
16.	404 (Not Found)	AU.CDR.Register.InvalidSoftwareProduct	Invalid Software Product	The software product requested provided to get the Data Recipient software statement assertion is invalid or cannot be found. Applies to: • “softwareProductId” path parameter for CDR Register APIs

#	HTTP Status Code	Error Code	Error Code Title	Error Scenario
17.	404 (Not Found)	AU.CDR.Resource.NotImplemented	Resource Not Implemented	The requested resource is part of the data standards but it is not implemented or not currently supported by the Resource Server. Reference to the specific resourceId or error encountered should be provided in the description field, without revealing sensitive information. Applies to: • Any unimplemented API
18.	404 (Not Found)	AU.CDR.Resource.NotFound	Resource Not Found	The requested resource is not part of the data standards and is not a holder-specific extension. Reference to the specific resourceId or error encountered should be provided in the description field, without revealing sensitive information. Applies to: • Invalid resource identifiers • URL resources that don't exist
19.	404 (Not Found) [in path] 422 (Unprocessable Entity) [in body]	AU.CDR.Resource.Invalid	Invalid Resource Identifier	The requested resource identifier is invalid, does not exist or will not be disclosed for business or security reasons. Reference to the specific resourceId should be provided in the description field. Applies to: • Resource identifiers as path parameters • Resource identifiers as request body parameters

#	HTTP Status Code	Error Code	Error Code Title	Error Scenario
20.	404 (Not Found) [in path] 422 (Unprocessable Entity) [in body]	AU.CDR.Resource.Unavailable	Resource Is Unavailable	The requested resource is currently in a state that makes it unavailable but this may change in the future. Reference to the specific resource Id should be provided in the description field. Applies to: • Resource identifiers as path parameters • Resource identifiers as request body parameters
21.	404 (Not Found) [in path] 422 (Unprocessable Entity) [in body]	AU.CDR.Resource.InvalidBankingAccount	Invalid Banking Account	The account bank account does not exist, the account is not associated to the consumer's active consent, or business reasons exclude executing this resource. Reference to the specific accountId should be provided in the description field. Applies to: • "accountId" path parameter • "accountId" request body parameters
22.	404 (Not Found) [in path] 422 (Unprocessable Entity) [in body]	AU.CDR.Resource.UnavailableBankingAccount	Banking Account Is Unavailable	The requested bank account is no longer associated to the active consent, a joint-account holder has withdrawn consent election or the account is currently in a state that makes it unavailable but this may change in the future. Reference to the specific accountId should be provided in the description field. Applies to: • "accountId" path parameter • "accountId" request body parameters

#	HTTP Status Code	Error Code	Error Code Title	Error Scenario
23.	406 (Not Acceptable)	AU.CDR.Unsupported.Version	Unsupported Version Requested	Either: • A request is made with a version that is less than the minimum version (x-min-v) the data holder supports. • A request is made with a version that is greater than the maximum version (x-v) the resource server supports. Both x-min-v and x-v are Positive Integers but the range between them is not a supported version. The data holder should respond with the highest supported version between x-min-v and x-v. If the value of x-min-v is equal to or higher than the value of x-v then the x-min-v header should be treated as absent. If all versions requested are not supported then the data holder should respond with a 406 Not Acceptable. Applies to: • "x-v" header parameter • "x-min-v" header parameter
24.	422 (Unprocessable Entity)	AU.CDR.Invalid.PageOutOfRange	Page Requested Is Out Of Range	Page out of range (e.g Valid pages are 5 and the client requested 10) Applies to: • "page" query parameter

#	HTTP Status Code	Error Code	Error Code Title	Error Scenario
25.	422 (Unprocessable Entity)	AU.CDR.Invalid.PageSizeTooLarge	Page Size Exceeded	Page size is greater than max (page_size > 1000) Applies to: • "page-size" query parameter
26.	422 (Unprocessable Entity)	AU.CDR.Invalid.Version	Invalid Version Requested	A request is made for a version that is not a positive integer. For example: • x-min-v or x-v are not Integers (e.g. x-min-v=foo, x-v=bar) • x-min-v or x-v are not Positive Integers (they are an Integer but <= 0) Applies to: • "x-v" header parameter • "x-min-v" header parameter
27.	418 (I'm A Teapot)	AU.CDR.IAmATeapot	I'm A Teapot	Client requests the status of their brewed coffee by calling GET /status/coffee. A resource server **may** return a 418, otherwise a 404 (Not Found) applies if the resource server doesn't offer coffee. Applies to: • GET /status/coffee

#	HTTP Status Code	Error Code	Error Code Title	Error Scenario
28.	503 (Service Unavailable)	AU.CDR.Service.Unavailable	Service Unavailable	A request is made but the API unavailable as part of a partial outage. The description should describe whether the outage is scheduled or unexpected and whether it is fully unavailable or partially unavailable. Applies to: • All APIs
29.	5xx	AU.CDR.UnexpectedError	Unexpected Error	An error code that can be used, when an unexpected error occurs. The resource server must populate the error description with a meaningful error description, without revealing sensitive information. Applies to: • All APIs

Current recommendation

The current recommendation of the Data Standards Body is to define an error catalogue within the data standards which lists the mandatory and recommended error codes that participants must use for API response. It is recommended that the catalogue documented in the previous section would form the initial basis of this catalogue within the standards.

It is also recommended that CDR error codes are not defined when errors are covered by normative references. This allows the normative standards to implement error handling as per the normative standards.

Of the options represented in the option section above this would include the adoption of:

- *Normative References Option 1 – Don't change normative references:* normative references will stand untouched to ensure there is a high level of conformance and out-of-the-box vendor support
- *Error Code Enumerations Catalogue Option 3 – Extend error catalogue to all CDR data flows:* a baseline error catalogue is presented in this decision proposal with further community consultation required to identify remain common error scenarios

Given there is only one core CDR Error Code currently defined in the data standards, this would result in a minor change to the data standards but create greater long-term predictability in the ecosystem. Addressing the issue of enhanced error handling now is most pragmatic because further delay will result in a greater implementation impact to the CDR. As more participants enter the ecosystem, the ability to make these changes will be reduced and overall consensus and conformance will be harder to reach.

The DSB is seeking feedback on this recommendation. In particular, feedback from data recipients on what granularity of detail they require for errors to be handled gracefully and programmatically in their software. Feedback on additions or amendments to the catalogue documented in the previous section is also specifically sought.

In addition, feedback is welcome on the implementation impacts of the changes recommended. Suggestions as to how the changes could be phased in safely for all participants are also welcome.

Appendix - Design Guidelines

The following guidelines are provided as design guardrails to frame and evaluate the options and outcomes.

Uniformity

1. Error handling must be defined consistently across the CDR Register and Data Standards.

a) The architecture should be consistently applied to the CDR Register, Data Recipients and Data Holders.

2. Error handling should observe and support normative references not override them.

a) Where normative references clearly define error responses, the data standards should not seek to override them to allow for conformant implementation of normative standards

Security

The security risk is the inadvertent exposure or inference of authentication information or sensitive data through the use of Error Codes. The most relevant OWASP risk is A3:2017 Sensitive Data Exposure, which is primarily controlled by the implementation of TLS. This proposal needs to take into consideration what information these Error Codes may reveal.

1. Error handling must not compromise the security of the Data Holder or CDR ecosystem.

a) The design process must identify which errors require an error message to be sent or displayed to the User.
b) Error Handling must be coded so that a failure of the Data Holder or any component of the Data Holder reveals only general information to a User. The Data holder may internally log more detailed technical information on the nature of the failure to a secure log in line with the Data Holder's usual and expected security and software management processes.
c) A User must not be able to use an error response to infer other information. For example, a message that a password is incorrect infers the username is correct.
d) A User must not be able to use an error response to gain elevated privileges.
e) A User must not be able to use an error response to gain access to a resource they are not entitled to access. For example, an attacker must not be able to employ API fuzzing techniques to infer a valid resource which belongs to another User.
f) Error data is for the purpose of machine to machine communications. Clients should interpret Resource Server errors but not display detailed descriptions verbatim to Users.

2. Error handling must not compromise the privacy of the consumer

a) The design process must identify PII data and exclude this from the detailed error message to observe data privacy and security concerns. For example, a customer's Internet Banking ID should not be revealed as an identifier for the customer.

b) Error Handling must be coded so that a failure does not reveal the Data Holder-side customer entitlements. For example, a message that the account is not available because there is a fraud lock on the account should not be shared.

Consumer Experience

Whilst improving machine-to-machine communication of errors is a key outcome, ultimately better servicing the consumer is the primary focus to ensure that the consumer is aware of errors, actions they can or should take, and the considerations of a consumer's situation when conveying errors to make sure their privacy and rights are protected.

1. Errors must be contextual to the user and the experience

a) For consumer-attended data requests and out-of-band notifications, a Client must convey errors that are comprehensible, clear and specific.
b) Errors must be described to consumers in the language of the Client, not the Data Holder. Clients should interpret errors by resource servers but not relay the errors verbatim because they have no control of consistency and content.
c) A Client should convey errors in the personality of the Client's brand so it familiar and consistent with the Client's other digital channels.
d) A Client should consider the context of the individual consumer and may personalise error messages appropriate to the consumer.

2. Errors must be meaningful to the consumer when consumers are asked to take action

a) Users must be informed and comprehend the choices they are asked to choose from when performing an action to resolve an error.
b) When actions need to be taken errors should be meaningful to the consumer, not the software.
c) When consumers are asked to retry actions or tasks, the Client should inform the consumer how many attempts will be tried.
d) Clients should inform consumers what alternative options they have when errors are encountered.
e) If a Client encounters an error that has material impact to the functioning of the use case and purpose but it cannot be resolved, the consumer should be notified (whether it is attended or unattended).

3. Errors must be sensitive to the circumstances of the related consumers

a) When a consumer is impacted by the exchange of an error, they should be made aware of the issue. For example, a joint account holder may be notified by text or email when they have withdrawn consent for an account they own but an ADR continues to request access to their data.
b) Errors must not divulge the situation of a Secondary Consumer to a Primary Consumer where it may cause harm. All consumers are protected by hardship rules and just because the primary consumer established consent, does not change a Data Holder's obligations to consider hardship of other consumers that jointly own an account related to the primary consumer's consent. For example, a victim of domestic violence should be protected even if they did not provide consent for the perpetrator to access their account data. Data Holders

must not leak information through error codes where it will knowingly cause harm or hardship.

Flexibility

No error code catalogue is ever complete. As new software products are developed new errors scenarios are defined. Likewise as the CDR is evolved to provide new data, different error scenarios are likely. A flexible architecture is enabling without imposing overhead on implementation.

1. Participants should have the flexibility to define new error codes

a) Participants—primarily Data Holders—must have flexibility where their processes identify new error scenarios within their typical development lifecycles. Because every data holder manages different data, has different constraints (e.g. which core banking system they use) and has different development release cycles, there needs to be enough flexibility for data holders to express new errors relevant for their software.
b) A flexible architecture should allow interfaces that integrate with both current and future components.

2. Custom participant error codes should not clash to avoid ambiguity across different participant implementations

a) With flexibility, there should still be structure to avoid unnecessarily creating more ambiguity.
b) The design process should provide a way that offers flexibility without causing collisions in the unreserved namespace.
c) The design process should prefer an approach that reserves namespaces and has a taxonomy that avoids the same error code meaning more than one thing across implementations.

3. The architecture should be adaptable to anticipated change

a) The architecture should provide a way that doesn't generate breaking change to all implementations and versions.
b) The architecture should adapt to the needs of participants. This includes new participants entering an industry, new industry sectors and new data being shared in existing or new APIs.

4. The architecture should allow participants extensibility to include meaningful custom data where it benefits their Users.

a) Participants must have the ability to define extensions to the Error data model in a way that doesn't break client implementations.
A good example where this may be useful is a Data Holder which defines an account opening API in the extensible, competitive space may need to convey additional detail when reporting errors that is not catered for by the core CDR error model.
b) The architecture should allow for a necessary level of descriptiveness to ensure errors are well understood by Users.

Supportability

1. The architecture should provide supportability for change over time

a) The design process should preference backwards compatibility because change over time with more participants in the ecosystem becomes harder.
b) The design process should define migration paths from custom error codes to common standard error codes that fosters adoption within pragmatic timeframes.
c) Changes to payload structure and data model should only include data requirements that are appropriate to convey the necessary level of detail required to support the ecosystem.

2. Custom error codes should be reviewed periodically for adoption as core CDR Error Codes

a) Custom error codes should be reviewed every six (6) months to identify new core CDR Error Code candidates
   Put simply, enhanced error handling should be reviewed on a regular basis (e.g. every 6 months) and new error scenarios identified and incorporated into the defined error catalogue.
b) Error codes should be adopted as core through a process of consultation and consensus where they benefit the ecosystem.
c) The design process should reduce ambiguity in the ecosystem where it leads to reduced costs, complexity, friction or ambiguity.

Appendix – Mapping of Error Scenarios from community issues

Error Code Mapping To Identified Error Scenarios

This table represents the mapping of proposed CDR Error Codes and HTTP Status Codes to the error scenarios identified by the community and consulted on via GitHub.

Source	Error Scenario	HTTP Status Code options	Proposed Error Code / HTTP Status Code
Issue #117	**Situation:** Account APIs called with one or more accountIds not part of consent. Some accountIds are part of consent. Does this API return "unauthorised" response, Or; does the API return data for only one account and not indicate that the response is partial? **Acceptance Criteria:** • Account API was called for 2 accounts specified in the request • Consent is valid • One account is a part of consent • The other one is not a part of consent	404 (Not Found) or 422 (Unprocessable Entity) or 200 (OK)	**HTTP Status Code** 404 (Not Found) [if path] 422 (Unprocessable Entity) [if request body] **CDR Error Code** AU.CDR.Resource.InvalidBankingAccount

Source	Error Scenario	HTTP Status Code options	Proposed Error Code / HTTP Status Code
Issue #117	**Situation:** Account APIs called with all accountIds not part of consent. All accountIds provided are not part of consent. Does the API return “unauthorised” response, Or; does the API return an empty data set? **Acceptance Criteria:** • Account API was called for 2 accounts specified in the request • Consent is valid • Both accounts are not part of consent	**Proposed:** 404 (Not Found) or 422 (Unprocessable Entity)	**HTTP Status Code** 404 (Not Found) [if path] 422 (Unprocessable Entity) [if request body] **CDR Error Code** AU.CDR.Resource.InvalidBankingAccount
Issue #117	**Situation:** Account APIs called with accountIds that are part of consent but cannot be shared because of data holder business rules. Includes account APIs requesting a single accountId. **Acceptance Criteria:** • Consent is valid • All Accounts are part of consent • Account cannot be shared because of an entitlements restriction	**Expected:** 404 (Not Found) or 422 (Unprocessable Entity) or 200 (OK)	**HTTP Status Code** 404 (Not Found) [if path] 422 (Unprocessable Entity) [if request body] **CDR Error Code** AU.CDR.Resource.InvalidBankingAccount [if sensitive] AU.CDR.Resource.UnavailableBankingAccount [if recoverable]

Source	Error Scenario	HTTP Status Code options	Proposed Error Code / HTTP Status Code
Issue #78	**Situation:** A request is malformed (because of say a badly formatted query parameter) and the requested page size is greater than 1000 records. **Acceptance Criteria:** • One or more input parameters is malformed • Page Size is greater than 1,000 (the maximum allowable page size)	Array of, or one of: 400 (Bad Request) or 422 (Unprocessable Entity) Implementation dependent with respect to error precedence.	**If Malformed encountered first:** **HTTP Status Code** 400 (Bad Request) **CDR Error Code** AU.CDR.Invalid.Field — **If Page Size encountered first:** **HTTP Status Code** 422 (Unprocessable Entity) **CDR Error Code** AU.CDR.Invalid.PageSizeTooLarge
Issue #174	**Situation:** Call to an unauthenticated resource API Call to Get Product Detail is made with a non-existent productId GET https://data.holder.com.au/cds-au/v1/banking/products/NonExistentProductID	404 (Not Found)	**HTTP Status Code** 404 (Not Found) **CDR Error Code** AU.CDR.Resource.NotFound

Source	Error Scenario	HTTP Status Code options	Proposed Error Code / HTTP Status Code
Issue #174	**Situation:** Call to an authenticated resource API. Call to Get Account Detail or Get Transaction Detail is made with a non-existent identifier. **Acceptance Criteria:** • Consent is valid • Account or Transaction ID does not exist	422 (Unprocessable Entity)	**HTTP Status Code** 404 (Not Found) [if path] 422 (Unprocessable Entity) [if request body] **CDR Error Code** AU.CDR.Resource.InvalidBankingAccount (if accountId) AU.CDR.Resource.InvalidResource (if any other resource)
Issue #118	**Situation** Status of the DR and or the software product changes but the DR continues to call the DH with a valid access token and the consumer's consent is still valid **Acceptance Criteria:** • ADR status (or Software Product Status) is not Active • Consumer consent is still active • DH has not yet withdrawn or cleaned up consent	403 (Forbidden)	**HTTP Status Code** 403 (Forbidden) **CDR Error Code** AU.CDR.Entitlements.InvalidAdrStatus AU.CDR.Entitlements.InvalidAdrSoftwareProductStatus
Issue #141	**Situation** An ADR requests transactions are filtered for an account by specifying a text filter when calling Get Transactions For Account. **Acceptance Criteria:** • ADR uses the "text" filter query • DH does not support text filtering	200 (OK)	**HTTP Status Code** 200 (OK) **CDR Error Code** N/A, isQueryParamUnsupported should be included in the meta object and set to true.

Source	Error Scenario	HTTP Status Code options	Proposed Error Code / HTTP Status Code
Issue #133	**Situation** An ADR requests an account resource where one or more accounts requested cannot be shared because it is in a frozen/suspended state by the DH. **Acceptance Criteria:** • Consent is valid • Account(s) are associated to consent • Account(s) have a Frozen/Suspended status.	200 (OK) or 422 (Unprocessable Entity)	**HTTP Status Code** 404 (Not Found) [if path] 422 (Unprocessable Entity) [if request body] **CDR Error Code** AU.CDR.Resource.InvalidBankingAccount
Issue #36	**Situation** GET request to /banking/accounts/{accountId} with an invalid accountID	404 (Not Found)	**HTTP Status Code** 404 (Not Found) **CDR Error Code** AU.CDR.Resource.InvalidBankingAccount
Issue 117	**Situation** Request to Get Accounts when consent has no associated available accounts	200 (OK) — with empty "accounts" list	**HTTP Status Code** 200 (OK) **CDR Error Code** N/A, empty “accounts” list returned
DP 68	**Situation** Brand Name is invalid: the "brand" query parameter is not a valid brand for the data holder	422 (Unprocessable Entity)	**HTTP Status Code** 400 (Bad Request) **CDR Error Code** AU.CDR.Invalid.Field

Source	Error Scenario	HTTP Status Code options	Proposed Error Code / HTTP Status Code
DP 68	**Situation** Product Category is Invalid: the "product-category" query parameter is not a valid product category	422 (Unprocessable Entity)	**HTTP Status Code** 400 (Bad Request) **CDR Error Code** AU.CDR.FieldType.Invalid.Field
DP 68	**Situation** Product Category is Invalid: the "product-category" query parameter is not offered by the data holder	422 (Unprocessable Entity)	**HTTP Status Code** 400 (Bad Request) **CDR Error Code** AU.CDR.Invalid.Field
DP 68	**Situation** Page out of range (e.g Valid pages are 5 and the client requested 10)	422 (Unprocessable Entity)	**HTTP Status Code** 422 (Unprocessable Entity) **CDR Error Code** AU.CDR.Invalid.PageOutOfRange
DP 68	**Situation** Page size is greater than max (page_size > 1000)	422 (Unprocessable Entity)	**HTTP Status Code** 422 (Unprocessable Entity) **CDR Error Code** AU.CDR.Invalid.PageSizeTooLarge
DP 68	**Situation** Page Number/Page size not a positive Integer	400 (Bad Request)	**HTTP Status Code** 400 (Bad Request) **CDR Error Code** AU.CDR.Invalid.PageSize

Source	Error Scenario	HTTP Status Code options	Proposed Error Code / HTTP Status Code
DP 68	**Situation** x-min-v or x-v are not an Integer (e.g. x-min-v=foo, x-v=bar)	422 (Unprocessable Entity)	**HTTP Status Code** 422 (Unprocessable Entity) **CDR Error Code** AU.CDR.Invalid.Version
DP 68	**Situation** x-min-v or x-v are not a Positive Integer (they are an Integer but <= 0)	422 (Unprocessable Entity)	**HTTP Status Code** 422 (Unprocessable Entity) **CDR Error Code** AU.CDR.Invalid.Version
DP 68	**Situation** Both x-min-v and x-v are Positive Integers but the range between them is not a supported version	406 (Not Acceptable)	**HTTP Status Code** 406 (Not Acceptable) **CDR Error Code** AU.CDR.Unsupported.Version
DP 68	**Situation** Valid version between x-min-v and x-v however x-v is not an implemented version. Data Holders supports x-min-v or higher however it does not support up to the requested x-v version	200 (OK) — Should respond using the highest supported version of the endpoint being called	**HTTP Status Code** 200 (OK) **CDR Error Code** N/A

Source	Error Scenario	HTTP Status Code options	Proposed Error Code / HTTP Status Code
DP 68	**Situation** Non existent Product ID (in get Product Details)	404 (Not Found)	**HTTP Status Code** 404 (Not Found) **CDR Error Code** AU.CDR.Resource.NotFound
DP 68	**Situation** Trailing / in a GET Products request	200 (OK) — (Ignore the trailing slash)	**HTTP Status Code** 200 (OK) **CDR Error Code** N/A
DP 68	**Situation** No matching products when filters/parameters are specified	200 (OK) — (0 Records)	**HTTP Status Code** 200 (OK) **CDR Error Code** N/A
DP 68	**Situation** Product ID is an Invalid ASCII String	400 (Bad Request)	**HTTP Status Code** 400 (Bad Request) **CDR Error Code** AU.CDR.Invalid.Field

Appendix: Issues seeking clarification on error handling

A number of conversation threads have been raised by the CDS community where the handling of errors, specific or general, are needed. These have been presented and summarised here:

Issue	Initial Post	Thematics	CDS-API-Stream Recommendations made (if any)	Link
API Error Schema Clarification	07 Nov 2019	• Define common CDR error codes • Error Code Mapping to Error Scenario • Error Code Mapping to Endpoint • When is "0001 – Account not able to be found" applicable	None	Issue #36
Enhanced error handling Stage 2 delivery risk impact assessment	07 Feb 2020	• Delivery risks for industry testing • CX needs to be considered • ADRs are most exposed to inconsistent error handling • When/if 404s are allowed • Flexibility for DHs should be maintained • Common Error codes should be defined • Consult on HTTP Status codes not defined in the standards (e.g. 404s)	None	Issue #118
Clarification on consent request scenarios	06 Feb 2020	How should consent entitlements be handled and the request of resource IDs that aren't part of consent	422 should be returned when an account Id is requested that is not part of consent	Issue #117

Issue	Initial Post	Thematics	CDS-API-Stream Recommendations made (if any)	Link
HTTP Header to be returned in the case where the request is not entirely well formed and a large page size is requested	18 Dec 2019	How should multiple errors be handled Is there a precedence or hierarchy of error handling?	None	Issue #78
HTTP Status codes supported by the specification	01 Apr 2020	When/if 404s are allowed	404 should be returned for PRD endpoints if there is no product for the given {productId}	Issue #174
Status code when a ADRs software product has become inactive	16 Apr 2020	How to handle ADR statuses kept in the CDR Register that are not active (e.g. suspended, revoked etc.)	403 should be returned if the ADR or the ADR Software Product is not "active"	Issue #188
List Transactions text filter isQueryParamUnsupported	22 Feb 2020	What should happen when an ADR requests a textual filtering of transactions but this functionality isn't supported by the DH?	DH should ignore filtering and response with a meta object including isQueryParamUnsupported=true	Issue #141
Account Status when Frozen	13 Feb 2020	How do DHs handle situations where an account is in a state that is neither OPEN or CLOSED (e.g. frozen or suspended)?	None	Issue #133
Using Duration standard for lastWeekDay in Scheduled Payments	05 Sep 2019	How should DHs handle invalid pagination	None	Issue #5

Issue	Initial Post	Thematics	CDS-API-Stream Recommendations made (if any)	Link
Error response code for x-v version headers	23 Mar 2020	How should DHs handle versions that are not supported or outside the allowable bounds for versions?	If a mandatory header is missing a Bad Request error is suitable. 400 Bad Request If an unsupported version is requested, such as a negative or non-integer value, a Not Acceptable error is suitable. 406 Not Acceptable	Issue #164
What's the expected behaviour? x-v set to a implemented version but a valid implementation version exists between x-min-v and x-v	07 Apr 2020	How should DHs handle versions that are not supported or outside the allowable bounds for versions?	406	Issue #179
Errors for Product Reference Data	29 Apr 2020	If GetProducts API receives invalid Brand or product-category query parameters in the request, is the expectation to respond with HTTP 200 with empty data {} or HTTP 400 bad request? If GetProducts API receives page param value out of range in the request, how does a data holder handle the behavior of pagination objects – links & meta for an empty response.	None	Issue #205

Identified Error Scenarios

Bendigo and Adelaide Bank Error Scenarios

This mapping of error codes was contributed by Bendigo and Adelaide Bank as part of their business analysis of HTTP Status Codes for publicly available PRD endpoints. It includes Bendigo's recommended HTTP status codes in each scenario.

#	Scenario	CBA	Westpac	NAB	ANZ	Bendigo's preferred Solution
1	Brand Name Invalid	400	400	200 (0 Records)	200 (0 Records)	400
2	Product Category is Invalid	400	400	400	400	400
3	Page out of range (e.g Valid pages are 5 and you enter 10)	200 (0 Records)	422	200 (0 Records)	200 (0 Records)	422
4	Page > 1000	200 (0 Records)	422	200 (0 Records)	200 (0 Records)	422
5	Page Number/Page size not a positive Integer	400	400	400	400	400
6	x-min-v not an Positive Integer	Could not get any response	Could not get any response	406	400	400
7	Non existent Product ID (in get Product Details)	200	404	400	403	404

#	Scenario	CBA	Westpac	NAB	ANZ	Bendigo's preferred Solution
8	Trailing / in a GET Products request	200 (same as GET Products without a trailing slash)	200 (same as GET Products without a trailing slash)	404	200 (same as GET Products without a trailing slash)	200 (Ignore the trailing slash)
9	No matching products when filters/parameters are specified	200 (0 Records)	200 (0 Records)	200 (0 Records)	200 (0 Records)	200 (0 Records)
10	x-v is a positive Integer but is not a supported version	406	406	406	406	406
11	Product ID is an Invalid ASCII String	200	400	400	403	400
12	Valid version between x-min-v and x-v however x-v is not an implemented version					200

Community submitted error scenarios

These error scenarios have been identified by the CDS community or encountered during industry testing as part of implementation either as a data recipient or data holder.

#	HTTP Status Code	Source	Error Scenario	Affected Resources
1.	**Expected:** 422 (Unprocessable Entity)	Issue #117	**Situation:** Account APIs called with one or more accountIds not part of consent. Some accountIds are part of consent. Does this API return “unauthorised” response, Or; does the API return data for only one account and not indicate that the response is partial? **Acceptance Criteria:** • Account API was called for 2 accounts specified in the request • Consent is valid • One account is a part of consent • The other one is not a part of consent	**Affected Resources:** • Get Balances For Specific Accounts • Get Account Balance • Get Account Detail • Get Transactions For Account • Get Transaction Detail • Get Direct Debits For Account • Get Direct Debits For Specific Accounts • Get Scheduled Payments for Account • Get Scheduled Payments For Specific Accounts
2.	**Proposed:** 422 (Unprocessable Entity)	Issue #117	**Situation:** Account APIs called with accountIds not part of consent. All accountIds provided are not part of consent. Does the API return “unauthorised” response, Or; does the API return an empty data set? **Acceptance Criteria:** • Account API was called for 2 accounts specified in the request • Consent is valid • Both accounts are not part of consent	**Affected Resources:** • Get Balances For Specific Accounts • Get Account Balance • Get Account Detail • Get Transactions For Account • Get Transaction Detail • Get Direct Debits For Account • Get Direct Debits For Specific Accounts • Get Scheduled Payments for Account • Get Scheduled Payments For Specific Accounts

#	HTTP Status Code	Source	Error Scenario	Affected Resources
3.	**Expected:** 200 (OK)	Issue #117	**Situation:** Account APIs called with accountIds that are part of consent but cannot be shared because of data holder business rules. Includes account APIs requesting a single accountId. **Acceptance Criteria:** • Consent is valid • All Accounts are part of consent • Account cannot be shared because of an entitlements restriction	**Affected Resources:** • Get Balances For Specific Accounts • Get Account Balance • Get Account Detail • Get Transactions For Account • Get Transaction Detail • Get Direct Debits For Account • Get Direct Debits For Specific Accounts • Get Scheduled Payments for Account • Get Scheduled Payments For Specific Accounts
4.	**Options Proposed:** • 400 (Bad Request) or • 422 (Unprocessable Entity) **Proposed:** • Implementation dependent	Issue #78	**Situation:** A request is malformed (because of say a badly formatted query parameter) and the requested page size is greater than 1000 records. **Acceptance Criteria:** • One or more input parameters is malformed • Page Size is greater than 1,000 (the maximum allowable page size)	**Affected Resources:** • All
5.	**Proposed:** 404 (Not Found)	Issue #174	**Situation:** Call to an unauthenticated resource API Call to Get Product Detail is made with a non-existent productId	**Affected Resources:** • Get Products • Get Product Detail.

#	HTTP Status Code	Source	Error Scenario	Affected Resources
			GET https://data.holder.com.au/cds-au/v1/banking/products/NonExistentProductID	
6.	**Proposed:** None provided	Issue #174	**Situation:** Call to an authenticated resource API. Call to Get Account Detail or Get Transaction Detail is made with a non-existent identifier. **Acceptance Criteria:** • Consent is valid • Account or Transaction ID does not exist	**Affected Resources:** • Get Account Detail • Get Account Balance • Get Transactions • Get Transaction Detail • Get Direct Debits For Account • Get Scheduled Payments For Account • Get Payees Detail • Delete CDR Arrangements
7.	**Options Proposed:** • 403 (Forbidden) or • 422 (Unprocessable Entity)	Issue #118	**Situation** Status of the DR and or the software product changes but the DR continues to call the DH with a valid access token and the consumer's consent is still valid **Acceptance Criteria:** • ADR status (or Software Product Status) is not Active • Consumer consent is still active • DH has not yet withdrawn or cleaned up consent	**Affected Resources:** • All authenticated APIs.
8.	**Options Proposed:** • 200 (OK) or • 400 (Bad Request) or	Issue #141	**Situation** An ADR requests transactions are filtered for an account by specifying a text filter when calling Get Transactions For Account.	**Affected Resources:** • Get Transactions For Account

#	HTTP Status Code	Source	Error Scenario	Affected Resources
	• 422 (Unprocessable Entity)		**Acceptance Criteria:** • ADR uses the "text" filter query • DH does not support text filtering	
9.	**Expected:** • 200 (OK) or • 422 (Unprocessable Entity)	Issue #133	**Situation** An ADR requests an account resource where one or more accounts requested cannot be shared because it is in a frozen/suspended state by the DH. **Acceptance Criteria:** • Consent is valid • Account(s) are associated to consent • Account(s) have a Frozen/Suspended status.	**Affected Resources:** • Get Account Detail • Get Account Balance • Get Transactions • Get Transaction Detail • Get Direct Debits For Account • Get Scheduled Payments For Account • Get Payees Detail
10.	**Expected:** • 422 (Unprocessable Entity)	Issue #36	**Situation** GET request to /banking/accounts/{accountId} with an invalid accountID	**Affected Resources:** • Get Account Detail
11.	**Expected:** • 200 (OK) with empty "accounts" list	Issue 117	**Situation** Request to Get Accounts when consent has no associated available accounts	**Affected Resources:** • Get Accounts
12.	**Proposed:** • 400 (Bad Request)	DP 68	**Situation** Brand Name is invalid: the "brand" query parameter is not a valid brand for the data holder	**Affected Resources:** • Get Products

#	HTTP Status Code	Source	Error Scenario	Affected Resources
13.	**Proposed:** • 400 (Bad Request)	DP 68	**Situation** Product Category is Invalid: the "product-category" query parameter is not a valid product category	**Affected Resources:** • Get Products • Get Accounts • Get Bulk Balances • Get Bulk Direct Debits • Get Scheduled Payments Bulk
14.	**Proposed:** • 400 (Bad Request)	DP 68	**Situation** Product Category is Invalid: the "product-category" query parameter is not offered by the data holder	**Affected Resources:** • Get Products • Get Accounts • Get Bulk Balances • Get Bulk Direct Debits • Get Scheduled Payments Bulk
15.	**Proposed:** • 422 (Unprocessable Entity)	DP 68	**Situation** Page out of range (e.g Valid pages are 5 and the client requested 10)	**Affected Resources:** • All
16.	**Proposed:** • 422 (Unprocessable Entity)	DP 68	**Situation** Page size is greater than max (page_size > 1000)	**Affected Resources:** • All
17.	**Proposed:** • 400 (Bad Request)	DP 68	**Situation** Page Number/Page size not a positive Integer	**Affected Resources:** • All
18.	**Proposed:**	DP 68	**Situation**	**Affected Resources:**

#	HTTP Status Code	Source	Error Scenario	Affected Resources
	• 400 (Bad Request)		x-min-v or x-v are not an Integer (e.g. x-min-v=foo, x-v=bar)	• All
19.	**Proposed:** • 400 (Bad Request)	DP 68	**Situation** x-min-v or x-v are not a Positive Integer (they are an Integer but <= 0)	**Affected Resources:** • All
20.	**Proposed:** • 406 (Not Acceptable)	DP 68	**Situation** Both x-min-v and x-v are Positive Integers but the range between them is not a supported version	**Affected Resources:** • All
21.	**Expected:** • 200 (OK) Should respond using the highest supported version of the endpoint being called	DP 68	**Situation** Valid version between x-min-v and x-v however x-v is not an implemented version. Data Holders supports x-min-v or higher however it does not support up to the requested x-v version	**Affected Resources:** • All
22.	**Expected:** • 404 (Not Found)	DP 68	**Situation** Non existent Product ID (in get Product Details)	**Affected Resources:** • Get Product Detail
23.	**Expected:** • 200 (OK) (Ignore the trailing slash)	DP 68	**Situation** Trailing / in a GET Products request	**Affected Resources:** • All

#	HTTP Status Code	Source	Error Scenario	Affected Resources
24.	**Expected:** • 200 (OK) (0 Records)	DP 68	**Situation** No matching products when filters/parameters are specified	**Affected Resources:** • All using query parameter filters other than the "text" filter
25.	**Proposed:** • 400 (Bad Request)	DP 68	**Situation** Product ID is an Invalid ASCII String	**Affected Resources:** • All