Data Standards Body
Technical Working Group

Decision Proposal 154 – Enhanced Error Handling – Error Code Payload Structure and Transition Arrangements

Contact: Mark Verstege
Publish Date: 22nd January 2021
Feedback Conclusion Date: 19th February 2021

Context

This is the third round of consultation on enhanced error handling. Consultation on enhanced error handling has been conducted across a series of decision proposals and community workshops. The purpose of this consultation has been to elicit feedback on improving the predictability and consistency of error handling between ADRs, Data Holders and the CDR Register.

This document is the primary consultation decision proposal based on feedback from a series of workshops and decision proposals. Two other decision proposals on enhanced error handling are published in this series. This series of decision proposal considers the feedback received from those consultations and is expected to be the final round of consultation before new standards are introduced.

To avoid duplication, this primary decision proposal provides the most context and background on the problem being solved whilst the remaining two dependent decision proposals focus on the specific changes proposed. The three decision proposals should be considered together however they have been intentionally split up so that one large decision proposal does not cover multiple solutions. The three decision proposals are as follows:

1. **Error payload structure (this document):** the proposed structure for error codes
2. **Error catalogue:** the proposed set of well-defined errors to be supported by data holders, ACCC Register and ADRs
3. **Discovery service**: the proposed mechanism for third-party clients to discover proprietary / custom data holder/ ADR error codes (i.e., where data holders support extensions to the CDS)

Document structure

This document includes four solution design sections

1. **A. Payload structure:** the proposed changes to the error response schema, incorporating previous community feedback
2. **B. Transition approach:** the proposed method to transition all data holders and ADR client implementations to the new structure and error catalogue in a way that allows for a transition period for all participants whilst allowing non-major data holders to go live with only standardised error responses.
3. **C. Response Versioning:** whether to version error responses and options to achieve it
4. **D. Implementation considerations:** the considerations given to implementation and transition for all participants

Background

Well-defined error handling is critical to the successful interoperability of Accredited Data Recipients (ADRs), Data Holders (DHs) and the CDR Register. As the Consumer Data Right (CDR) ecosystem grows — soon to accommodate all Accredited Deposit-taking Institutions (ADIs) — the complexity of integrating across a many-to-many system becomes more challenging. Without a well-defined set of standardised error responses, ADRs are forced to build out conditional logic to process each DH differently, possibly inconsistently. This is a non-trivial effort for ADRs that is difficult to scale.

Without standards to provide deterministic error handling, this will lead to inconsistent handling of errors by ADRs and a poorer consumer experience that may not only be confusing but result in interoperability issues that impacts consumer experience.

To provide predictable and scalable error handling across the ecosystem, this decision proposal proposes payload changes to support a standardised error catalogue that participants can reliably and consistently implement to.

This decision proposal assumes prior knowledge of the previous consultations and decision proposal series.

Moving beyond HTTP status codes

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 implement can improve consistency and reduce the complexity cost for API clients: especially ADRs.

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

Consultation conducted

As a problem space, Enhanced Error Handling has been consulted on over a series of Decision Proposals and workshops to agree on the structural requirements. This proposal consolidates technical feedback across the following decision proposals:

- 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
- 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

- Define the structural data standards required to support error handling
- Define the transition path towards standardised CDR error codes for both ADRs and DHs
- Determine what versioning strategy is required for error codes

Feedback Provided

Feedback has been received via four channels:

1. GitHub decision proposals
2. DSB facilitated workshops
3. CDR Implementation Calls
4. ZenDesk community questions

The workshop outputs are available here:

- **Workshop 1 – Payload Data Structure:** https://miro.com/app/board/o9J_kr5jpHQ=/
- **Workshop 2 – Error Catalogue:** https://miro.com/app/board/o9J_krfCaQQ=/
- **Workshop 3 – Consumer Experience:** https://miro.com/app/board/o9J_knxw6Pc=/ (input) and https://miro.com/app/board/o9J_khmXIzU=/ (findings)

The feedback has been summarised in each solution design section.

Key principles and considerations

To accommodate a smooth transition and a flexible, extensible solution, a few key considerations are important when considering the solution options:

1. **Error codes should be uniformly defined and implemented across all participants:** there must be uniformity across the CDR Register and Data Standards for ADRs and Data Holders
2. **Return standardised error codes before custom error codes:** participants must use CDR error codes where they are identified instead of creating a custom error code that represents the same error scenario
3. **Errors must be meaningful to API clients and consumers:** Error codes should express sufficient context to clients and distinguish different errors with sufficient granularity to enable clients to adequately handle commonly encountered types of errors.
4. **Error codes must be defined in a way to be supportable across a large ecosystem of participants:** the list of errors should not be so large or difficult to implement that it cannot scale. The error codes must be applicable to general situations across implementations and sectors.
5. **The error schema needs to support a phased transition plan:** transition between existing responses and standardised error responses requires mapping from code/title/status.
6. **Participant-specific error codes must be mapped to a respective standardised CDR error code:** Error responses must provide sufficient detail of previous custom error codes to ensure clients can adequately map to standardised error responses.
7. **Error codes should be extensible for implementation specific errors:** participants will continue to support implementation specific error codes. A good example where this will frequently occur is voluntary data sets. Where data holders choose to develop extensions to the required CDR data set – say new APIs for insights or payment initiation which may be provided under a commercial license – would give rise to implementation specific error responses.
8. **The CDS defines a set of baseline error codes:** having a baseline error code that the participant can extend allows all clients to handle the generic error whilst specialising their error handling for any participant specific implementation codes without causing breaking client implementations.
9. **Error codes must be uniquely identifiable** to avoid ambiguity and clashing across a reserved enumeration set.

10. **Error responses must express the appropriate context:** Error codes must describe the problem encountered with sufficient detail without disclosing sensitive business context or security context.
11. **Error response mustn't share the customer's context:** In line with principle 10, error responses must consider the consumer's privacy and circumstances and not divulge sensitive information to the client.
12. **Error codes are defined as URNs:** URNs are defined according to RFC 8141 for interoperability and conformance.
13. **Error codes and Error titles are case insensitive:** in line with principle 9 and principle 12, error codes should be treated as case insensitive.

A. Payload structure

Overview

Feedback from participants determined that a URN format was preferred over other structured formats.

Note:

The DSB is seeking to define the CDR URN specifiers within a national URN namespace. Work is underway to register an RFC with IANA to formally recognise a top-level namespace identifier. The purpose of this approach is to ensure the URN structure is defined under a new, nationally relevant, and internationally recognised, namespace under which, both related CDR URN specifiers as well as other national data standards URN specifiers may live.

Furthermore, this will ensure the formally recognised namespace identifier cannot be claimed or used for other purposes. Much like domain names, after registration, it precludes the use of a namespace identifier under the CDR. This timeframe is expected to be longer than the timeframe of transition obligations.

The DSB has designed the URN structure for error codes in a way that it would be possible to move to the formal namespace when ratified with minimal effort for participants. Until the time that the formal URN namespace can be registered, the CDR will adopt an informal namespace "AU-CDS".

Feedback provided

Decision Proposal 119 considered the necessary payload data structure changes to be made. Based on feedback to the proposal, a well-defined enumeration structure for error codes is supported with changes to the error code payload to support transition from custom error codes and a simplified mechanism of discovering any custom codes a data holder supports via data holder developer portals.

Feedback has been summarised below:

- **URN format should be used for the error code:** this provides consistency and software libraries exist to parse and produce URNs
- **Use of a deterministic structure for error codes:** conditional categorisation or segmentation makes it difficult to parse in a consistent and well-defined manner. Alternatively, a strictly defined structure ensures there are

- **Payload structure must support customisations:** or participant specific error codes where a CDR error code is not defined (such as voluntary extensions to the CDR APIs)
- **Allow the use of the JSONAPI source/pointer parameters:** this should be allowed but at the discretion of the implementation rather than a mandatory requirement. This will reduce breaking changes required in future for write access
- **Error codes should not be case sensitive:** to improve support, error codes should be case insensitive
- **Define generic "catch-all" error codes:** to ensure there is always a standard CDR error code that can accommodate custom error codes, generic "catch-all" error codes should be defined which custom error codes extend.

URN Structure

```
urn-string = "urn:" NID ":" metatype ":" sub-type ":" error-category "/"
error-code [ "?=" q-component ]

    NID       = "au-cds"

    metatype = "error"

    sub-type = cds-all / cds-register / cds-banking / cds-energy

        cds-all      = "cds-all" string. An error code common to all
                       API responses,

        cds-register = "cds-register" string. Reserved for CDR Register
                       issued error codes only,

        cds-banking  = "cds-banking" string. An error code specific to
                       the CDR banking APIs only,

        cds-energy   = "cds-energy" string. An error code specific to
                       the CDR energy APIs only.

    error-category = string. The high-level category code for the error
                     defined in the CDR Error Catalogue

    error-code     = string. The specific error encountered, defined in
                     the CDR Error Catalogue
    q-component    = The q-component is intended for passing custom
                     parameters to either the named resource or a
                     system that can supply the requested service, for
                     interpretation by that resource or system.
                     They are analogous to query components from a REST
                     API perspective.
```

Options identified

These options consider how participants can publish implementation specific error codes that extend the standardised error codes.

Three new fields to support implementation specific errors are defined as follows:

xStatus: An application-specific HTTP status code where it differs to the standard CDR error code. This is useful for mapping legacy handling of error codes.

xCode: An application-specific error code, expressed as a string value, specific to the participant.

xTitle: a short, human-readable summary of the problem that **SHOULD NOT** change from occurrence to occurrence of the problem, except for purposes of localization, specific to the participant.

Option A1:Custom error codes extend the URN string

Participants use a base CDR error code and define any custom error codes using three new fields as query components (q-component).

Examples:

1) `urn:au-cds:error:cds-all:Header/InvalidVersion?=xStatus=400&xCode=ERR_0406&xTitle=Version+5+Is+Not+Acceptable`
2) `urn:au-cds:error:cdr-all:Field/Invalid?=xStatus=400&xCode=ERR091&xTitle=Invalid+Product+Category+Filter`
3) `urn:au-cds:error:au-banking:Action/Invalid?=xStatus=422&xCode=WBC-PI010&xTitle=Insufficient+funds`

Option A2:Custom error codes as additional fields in ResponseErrorList_errors

Participants use a base CDR error code and define any custom error codes using two new fields in the "meta" object.

This option makes changes to the ResponseErrorList_errors to allow participants to define any custom error handling. Custom errors must extend a base error code.

Examples:

```
"errors": [
  {
    "code": "urn:au-cds:error:cdr-all:Header/InvalidVersion",
    "title": "Invalid Version",
    "detail": "Version 5 was supplied but Get Product Detail only supports
               x-min-v = 2, x-v = 3",
    "status": "406",
    "meta": {
      "xStatus": "400",
      "xCode": "ERR_0406",
      "xTitle": "Version 5 Is Not Acceptable"
    }
  }
]
```

B. Transition approach

Transition arrangements need to consider the phasing in of DH obligations and a period of deprecation after any mandatory obligation dates for DHs commence. This ensures that DHs have sufficient time to implement the changes and ADRs have a stable period of transition before retirement of any custom error codes.

Regarding all possible transition options, a few prevailing principles are considered:

- Allow DHs to support standardised error codes as early as possible
- Allow ADRs to discover the mapping of old and new error codes over a sufficient transition period

- Enable DHs to continue describing their custom codes, if desired, as a specialisation of standardised error codes
- Where possible, reduce impact to ADRs
- Allow ADRs to discover changes in HTTP status codes during the period of transition (e.g., a 422 will standardise to become a 404 for a given DH implementation)
- ADRs require a period of transition *after* mandatory obligation dates for DHs before deprecation and retirement of DH specific custom error codes

Options identified

Option B1: Return both old and new error codes during transition (organic transition)

This option would return both old and new error codes for an agreed period of time. This would allow ADRs to adapt their client software to transition from their current error handling logic to the standardised error handling. Although this places some extra burden on Data Holders, the objective is to minimise impacts to clients whilst enabling Data Holders to stage their delivery of the new error responses.

Under this option, data holders would individually choose how they phase in implementation of standardised error codes within their existing delivery schedules and priorities.

Considerations

- Allows for a gradual transition for clients to map old and new error codes
- Assists data holders to phase in implementation of standardised error codes (not all error responses need to switch over immediately)
- Supports more stable change management for each participant and the ecosystem as a whole

Option B2: Return both old and new error codes during transition (transition tranches)

Similar to "Option B1: Return both old and new error codes during transition (organic transition)" however a set of designated tranches are defined to stagger the transition. This would require ADRs and DHs to transition a set of end points in stages then another set of end points and so on.

Considerations

- Allows for a gradual transition for clients to map old and new error codes
- Supports stable change management for each participant and the ecosystem as a whole
- May require more work and customisation for DHs where errors are responded by API Gateways etc, involving conditional logic where the same error code must now be responded as either old or new based on the end point being called.

Option B3: Cutover to new error codes on a fixed obligation date

Old error codes are described using the implementation specific error code fields using the payload conventions listed above. In this option, there is a fixed cutover date without a period of transition. This allows DHs to implement the new standardised error codes whilst dropping support for customer error codes.

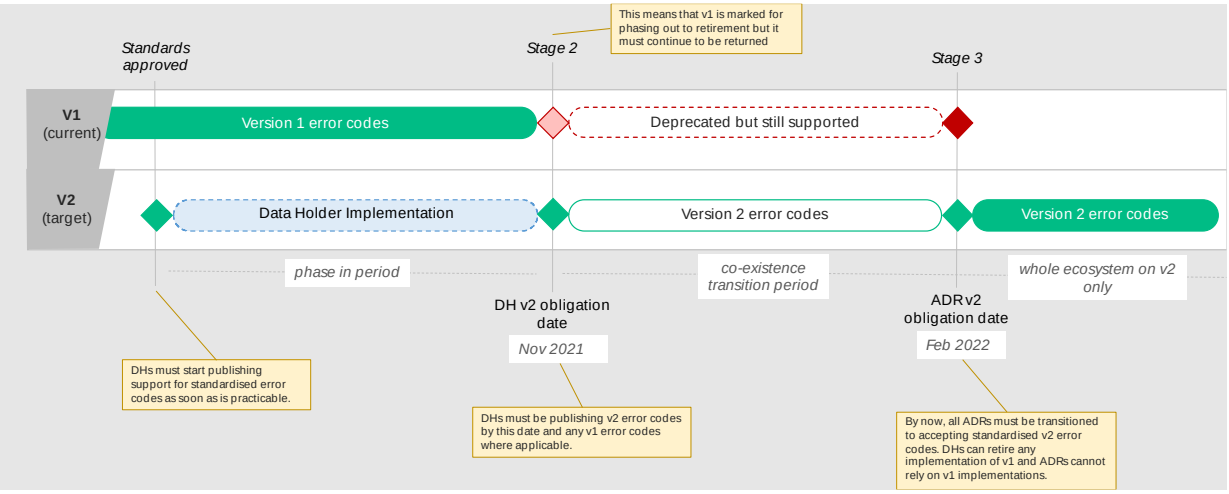
Considerations

- This option doesn't allow clients to progressively discover the mapping between old and new error codes.
- It also imposes a more brittle change management process on the regime whereby all data holders and the CDR Register must switch to the new error code format on a fixed date. All clients must then cater for the new error codes being returned from that obligation date.
- Whilst this may allow the client to negotiate the version of the error, they can consumer, it does not assist the client to know what the new error code will be once mandatory cutover occurs.

Transition timeframe

If " Option B1: Return both old and new error codes during transition (organic transition)" is adopted, all DHs may return conformant error codes as soon as is possible. It is proposed that Data Holders must support the new error codes no later than November 2021.

Old error codes can be retired from February 2022. At a high level, the transition is proposed as follows:



C. Response Versioning

Option C1: Version all end points

This option treats a change in the error schema to be a breaking change to all resource end points. Phasing in of error codes also requires Data Holders to support two versions of each API end point during the transition period.

Considerations

- This implies any future breaking changes to the error
- The client can negotiate a different end point version based on the error response
- Adding new error code URNs would not result in a breaking change in future
- Any breaking changes to the "happy path" response payload will result in another version of affected end points implying Data Holders may be required to support three or more versions of some end points.
- This increases the implementation and operational cost to participants.

Option C2: No end point versioning

At present, each resource end point in the data standards is versioned where breaking changes to the API interface incrementing the version of the end point. This allows client and

server to negotiate the version of the end point being called. Error responses are not currently versioned and changing the error response is considered as a non-standard end point response which should not result in a versioning of every end point.

Considerations

- This option relies on a managed transition across the ecosystem defined in "B Transition approach"
- This implies error responses are transitioned over a period of time but not explicitly versioned
- A client cannot negotiate a different end point version based on the error response
- Adding new error code URNs would not result in a breaking change in future

Option C3: Use version negotiation for error codes

In this option, error response versioning request headers are introduced. Similar to end point versioning, a minimum and target error response version are supplied by the ADR. If none are supplied, then the version defaults to the minimum supported by the data holder. In this situation it allows each ADR and DH to negotiate the version of the error response to be returned.

Considerations

- This option doesn't allow clients to progressively discover the mapping between old and new error codes.
- Whilst this may allow the client to negotiate the version of the error, they can consumer, it does not assist the client to know what the new error code will be once mandatory cutover occurs.
- Likely to add more work to both ADR and DH build

D. Implementation considerations

Implementation consideration has been given to how both ADRs and DHs can migrate to the target state.

Primarily, this has involved factoring into account three main considerations:

1. Providing time for Data Holders to make changes to their existing implementations
2. A way for ADRs to determine the mapping of old and new error codes for each DH
3. Ensuring there is sufficient time for ADRs to cutover to handling the new error codes

Further to this, care has been given to ensure the target state makes it easier for participants to adopt new error codes in future without change to the data model. It is envisaged that a periodic review process would consider adoption of any new standardised error codes identified by the community.

The aim has been to ensure any DH entering the ecosystem can, if desired, adopt the new standardised error codes from July 2021. Where possible this reduces ongoing maintenance to affected DHs. It is recognised that this will not be possible for all current or future DHs and allowing sufficient time for the cutover will be required to ensure DHs can deliver existing implementation obligations whilst providing certainty to ADRs they can continue to handle errors in a reliable way.

Impacted APIs

This change impacts Data Holder APIs for product reference data (PRD) and consumer data sharing APIs and the CDR Register APIs.

URN Versioning

URNs include a version prefix to accommodate any breaking changes to a namespace specific string.

Initial Data Holders

Initial data holders will be impacted. They will require time to transition from custom error codes to standardised error codes.

Reciprocal Data Holders

Reciprocal data holders go live with consumer data sharing APIs no later than March 2021. It is likely that their implementations will go live with custom error codes and require a period of transition to standardised error codes.

All Other Data Holders

Non-major ADIs go live with consumer data sharing APIs from July 2021. Any DHs voluntarily going live prior to July 2021 may consider their implementation to be compliant with the new error codes from commencement.

It is possible that non-major ADIs will implement custom error codes whilst others will be able to leverage standardised error codes in line with commencement of their consumer data sharing APIs.

Early Adopter Data Holders

Any DHs that go live with **only** the standardised error codes would not require any future retirement of old error codes. This would reduce the work required for these DHs to support two versions.

ADRs

Any ADRs connected to data holders producing custom error codes will be impacted and require a period of transition to update their client applications to learn then expect the standardised error codes in place of each's data holder custom codes.

Where standardised error codes suggest a MAY or SHOULD, there is still the possibility that ADRs must continue to accommodate implementation nuances from each data holder, but this effort should be reduced.

Public Clients

Any public clients collecting PRD data will be impacted. Similar to ADRs, any public clients must be updated to handle the new standardised error codes. The collection of PRD data by public clients is outside the scope of the decision proposal.

CDR Register

The CDR Register will be impacted when collecting metrics data from Data Holders

As a single client, with a limited set of errors to handle when collecting metrics data, it is considered there is sufficient time to transition.

The CDR Register will also be impacted with build to update any errors produced by the CDR Register such that they also leverage the standardised set of error codes.

Current recommendation

Error response schema

It is recommended that the error response payload be updated to represent error codes as URNs with the changes to the error response schema defined in "Option A2: Custom error codes as additional fields in ResponseErrorList errors".

Transition arrangement

It is recommended that the method of transition "Option B1: Return both old and new error codes during transition (organic transition)" be adopted.

Response versioning

It is recommended that "Option C2: No end point versioning". Whilst there are trade-offs with this approach it is seen as less disruptive than versioning all public and authenticated endpoints.

Obligation dates

Feedback is welcomed on obligation dates. Ideally, the DSB would like to encourage early adoption of error codes with a period of support that enables DHs to adopt these changes with sufficient lead time and allows ADRs to update their client software to discover these changes and switch over.

To this effect, it is proposed that DHs can begin transition as soon as is practicable and the obligation dates are as follows.

Stage 1 (current error responses)	**Stage 2** (transition period and co-existence of error codes)	**Stage 3** (DHs only support standardised error codes)
As soon as possible	November 2021 Data Holders MUST support standardised error codes by this date and continue to support any custom error codes	February 2022 Data Holders may deprecate any custom error codes

The DSB is seeking feedback on these recommendations in conjunction with the dependent decision proposals for the error catalogue and discovery service.

Feedback is welcome on the implementation impacts of the changes recommended and the transition timeframes.

Standards changes to make

The standards changes proposed in this document include:

1. **Error response schema:** changes to the ResponseErrorList error response schema including the URN error code format.

    a. Change "error > code" to be an enumerated type of standardised error code URN strings
    b. Add "xStatus", "xCode", "xTitle", "xDescription" support in the "error > meta" object response for transition arrangements so Data Holders can support legacy error codes

The complete changes to the error response structure are provided in "Appendix A - Error response schema".

Appendix A - Error response schema

New changes are highlighted in green.

```
"ResponseErrorList_errors" : {
      "required" : [ "code", "detail", "title" ],
      "x-conditional": ["status"],
      "properties" : {
        "code" : { ...truncated for brevity... },
        "title" : { ...truncated for brevity... },
        "detail" : { ...truncated for brevity... },
        "status" : {
          "type" : "string",
          "description" : "The HTTP status code applicable to this problem,
expressed as a string value. MUST be provided."
        },
        "meta" : {
          "type" : "object",
          "description" : "Optional additional data for specific error
types",
          "properties" : {
             "xStatus": {
                "type": "string",
                "description": "The application-specific HTTP status code
applicable to this problem, where it differs to the standardised error
code, expressed as a string value"
             },
             "xCode": {
                "type": "string",
                "description": "An application-specific error code,
expressed as a string value, specific to the participant"
             },
             "xTitle": {
                "type": "string",
                "description": "a short, human-readable summary of the
problem that SHOULD NOT change from occurrence to occurrence of the
problem, except for purposes of localization, specific to the participant"
             }
          }
        }
      }
    }
```