

ERROR MAPPING 4.5

# Document Status

status: Deprecated (valid values are < Request for Comment, Preliminary Review, Public Review, Architectural Review, Final Review, Published, Deprecated)

# Change Log

|  |  |  |
| --- | --- | --- |
| Version |  | Changes |
| 0.1 |  | * Initial Creation |
| 0.2 |  | * Added Validation Error Codes |
| 0.3 |  | * Added/modified generic error codes with parameterized detail text to support specific data integrity/validation issues and actionable error messages. |
| 0.4 |  | * Convert to XSD format and reference in this document. Update example errors. |
| 0.0.5 |  | * Change plural items to List form |
| 0.0.6 |  | * Transmission of errors |
| 0.0.7 |  | * Updated Overview of Specification |
| 0.0.8 |  | * Added Error Types |
| 0.0.9 |  | * Make language support up to end points * Add substitution ability for portions of the error message |
| 3.0 | **10/29/2013** | * Versioning and format change with release CUFX 3.0 |
| 3.0 | **12/18/2013** | * Minor formatting and text updates |
| 3.1 | **07/17/2015** | * Updated to release 3.1 |
| 3.2 | **05/10/2016** | * Updated to release 3.2 |
| 3.3 | **02/15/2017** | * Updated to release 3.3 |
| 4.0 | **02/19/2018** | * Updated to release 4.0, Microsoft Global bug fix, Use of Error.xsd is deprecated. All messaging support for success, warning, information and error messages have been moved into MessageContext. All error messages have been refactored into a textual response. Duplicated and or overlapping error messages were removed to provide a consistent messaging response mechanism. |
| 4.1 | **12/10/2018** | * Updated to release 4.1, Use of Error.xsd is deprecated. |
| 4.2 | **03/05/2019** | * Updated to release 4.2, Use of Error.xsd is deprecated. \*\*\* Release 4.2 is a breaking fix release. \*\*\* Errors found in App, ArtifactFilter, and BillFilter required a breaking fix to align with the standard and prevent additional implementation difficulties going forward. |
| 4.3 | **10/07/2019** | * Updated to release 4.3, Use of Error.xsd is deprecated. renamed file removing version as proper version control is being used in Github. |
| 4.4 | **10/20/2020** | * Updated to release 4.4, Use of Error.xsd is deprecated. |
| 4.5 | **04/02/2021** | * Updated to release 4.5 |

# Overview of Specification

The CUFX Error Mapping specification describes the standardize error codes and format for the CUFX specifications. It contains all the specific error messages and their customizable constructs to provide additional detail for the end user and client application.

# Any known Errors in the document

|  |  |
| --- | --- |
| **Error Description** | Status of Error |
|  |  |

Table of Contents

[Document Status 1](#_Toc68098993)

[Change Log 1](#_Toc68098994)

[Overview of Specification 2](#_Toc68098995)

[Any known Errors in the document 2](#_Toc68098996)

[Document Conventions 3](#_Toc68098997)

[CUFX API and Documentation Support 3](#_Toc68098998)

[Release 4.0 Global Update Notes 3](#_Toc68098999)

[Release 4.4 Global Update Notes 4](#_Toc68099000)

[Concepts 4](#_Toc68099001)

[Data Elements – MessageContext 5](#_Toc68099002)

[Error Response Example - REST-JSON 5](#_Toc68099003)

[StatusType 6](#_Toc68099004)

[StatusCode 6](#_Toc68099005)

[Data Elements – errorlist – Deprecated 6](#_Toc68099006)

[Error Response Example - REST-JSON 6](#_Toc68099007)

[Error codes 7](#_Toc68099008)

[Error types 7](#_Toc68099009)

[Bibliography 7](#_Toc68099010)

# Document Conventions

List any document conventions such as what bold and italics mean and how the document is intended to be read.

“Within this specification, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in W3 Working Group (W3C)]. However, for readability, these words do not appear in all uppercase letters in this specification.

At times, this specification recommends good practice for authors and user agents. These recommendations are not normative and conformance with this specification does not depend on their realization. These recommendations contain the expression "We recommend ...", "This specification recommends ...", or some similar wording.”

All formatting in this document utilize Word Styles.

All Citations must utilize Word Citations to automatically show at the end of the document.

All updates after the initial creation must be performed using Tracking Changes turn on and Accepted by the Architecture committee

# CUFX API and Documentation Support

CUFX is published to SwaggerHub at <https://app.swaggerhub.com/apis/dlacroix01/CUFX> . The latest default version will load automatically.

The purpose of this publication is to demonstrate the full range of CUFX messaging. Provide for complete documentation of the entire message structure and example usage.

Swaggerhub as a normal support feature also translates into several dozen of the most common and in demand client / server formats. This provides a technology specific version of the CUFX standard for essentially all platforms.

The CUFX Specification documents have been written to provide for limited examples of case usage but do not reflect the entirety of elements present in any given message. Please see the latest version of CUFX in Swaggerhub for the complete message and for superior documentation support.

# Release 4.0 Global Update Notes

CUFX Release 4.0 introduces a number modifications that significantly improves the standard and is not backward compatible with prior versions.

Messaging paradigm shift. Prior to CUFX 4.0 a Message Object would be sent and would expect the Object List to be returned or the error message. The response had to be interrogated to determine what was received. With CUFX 4.0, the Object Message that is sent is also expected to be the Object that is returned. Significant improvements have been made to the Message Context to fully support Success, Informational, Warnings and Error responses. End Points may continue to use the prior methods, but use of the Error.xsd is depreciated; all functionality has transitioned into MessageContext.xsd.

Date Range Filtering. A global update was applied across the standard to remove the pairs of date filter elements for any given range and replaced with a single Common.xsd definition DateRange complex type. This makes date range filtering completely uniform across the standard and associates the startDateTime and endDateTime together as an object set.

As example: elements transactionStartDateTime and transactionEndDateTime were replaced in the AccountFilter.xsd with transactionDateRange.

Microsoft Serialization Bug. We discovered the root cause of a serialization error impacting CUFX. A known Microsoft Serialization error from 2006 is present for single element complex types. It causes a naming error of the serialized constructs. If both endpoints are using a Microsoft compilation the error is consistent and does not present itself, the names are both wrong but pass data successfully. When one end point is not using a Microsoft compilation, the field names are in variance and fails. If both end points are using non-Microsoft compilation the serialization would be correct and match.

CUFX 4.0 has applied a global update across all list types throughout the standard. The CUFX list construct was consistently a single element complex type. For all occurrences we have applied an extension base of common:ListBase. ListBase provides pagination support and also resolves the Microsoft serialization error. No longer being a single element complex type, Microsoft compilation now generates the correct names. This will necessitate prior (Microsoft) implementations to remap to the correct serialized names.

# Release 4.4 Global Update Notes

CUFX Release 4.4 introduces a significant enhancement for complex Account identification and filter navigation. The foundational architectural design premise for account navigation is that the CUFX AccountId would be a unique value unto itself within a given institution, or that a composite unique key would be passed. With the direct support of several core system providers it was established that that later case is predominate. The AccountId is generally not a unique value unless in combination with several other values such as AccountType and AccountSubType. Hence passing a unique AccountId meant that the organization had to overload the element value making filtering implementation specific and forcing the endpoints to map overlay the accountId to unpack the value.

CUFX now fully recognizes this architectural paradigm while continuing to support the original architecture.

A global update was applied to provide the elements accountType and accountSubType in all objects that contained accountId.

A new filter list – AccountIdentificationList has been added to all account related filters so that AccountId, AccountType, and AccountSubType can be structured properly for discreet filtering of complex account keys, support filtering by the sub keys and also support inbound and outbound account filtering using the accountToFromIndicator.

By expanding the architectural paradigm to support a non-unique AccountId CUFX is now positioned to better support core adoption of the standard.

# Concepts

The CUFX Data Provider sends error messages when the client fails to provide an expected request or the Data Provider has an internal failure. Each error response consists of a non-200 HTTP response roughly corresponding to the accepted list of HTTP status codes and a payload containing a detailed error code and message. Fields not returned should be assumed to be null. The header record of the request contains the language that the error message should be displayed in. It is up to the endpoints to determine if multi-language will be supported.

\*\*\*\* Depreciated <start>

*The CUFX service can return default errors like HTTP error code 500 or 404. When a CUFX Data Provider returns those error codes, it’s the responsibility of the service consumer to look deep into the error object for more details.*

*The CUFX service needs to configure to return the error object when a HTTP error code is returned.*

*Example:*

*If you build using .net platform this behavior can be overridden via web.config file as depicted below. This will not trash the custom error object that’s part of the response when a HTTP error code is returned by the CUFX service.*

*<configuration>*

*<system.webServer>*

*<!-- ensures that on remote server we get our custom json responses for HTTP code 500 and NOT generic IIS response-->*

*<httpErrors existingResponse="PassThrough"></httpErrors>*

...

\*\*\*\* Depreciated <end>

Starting in CUFX release 4.0 the object message that is sent to an end point is expected to be the object that is returned from the end point. MessageContext has been expanded to provide full messaging support for all success, warning, informational, and error returns. This means the sending endpoint will not have to interrogate the response message to determine if it the object list or the error object.

Example: if the end point sends an accountMessage object, an accountMessage object will be returned. One or more status codes will be present in the StatusList of MessageContext. The status list can contain any number of success, warning, informational, and error status that may have resulted from processing the message. The successful message response will contain the returned records in accountList.

For CUFX 4.0 Error.xsd will continue to be available to support prior implementations, but is deprecated. No additional support will be extended. See messageContext for usage and all future updates.

# Data Elements – MessageContext

The data elements returned by an error are contained messageContext statusList

### Error Response Example - REST-JSON

"messageContext": {

"cufxVersion": "4.5.0",

"requestId": "requestId1",

"vendorId": "vendorId1",

"appId": "appId1",

"fiId": "fiId1",

"dataSourceId": "dataSourceId1",

},

"statusList": [

{"status": {

"statusType": "Error",

"statusCode": "ValueOutOfRangeError"

}

},

{"status": {

"statusType": "Informational",

"statusMessage": "Expected value in the range of A-Z, 0-9"

}

}

]

…

### StatusType

**Success** - The message associated with this response type was a success.

**Informational** - Additional information about the response, including verbose responses.

**Warning** - Informational warning that should not stop processing.

**Error** - The message is associated with response was an error.

### StatusCode

See the MessageContect.xsd StatusCode simple type for all enumeration.

# Data Elements – errorlist – Deprecated

The data elements returned by an error object is contained in Error.xsd. This method is deprecated.

### Error Response Example - REST-JSON

{

“errorList”: [

{

“code”: 499,

“type”: 400,

“subCode”: 548541521,

“message”: “General Error. Review subCode for more Information”

},

{

“code”: 400,

“type”: 300,

“subCode”: 0,

“message”: “Bad Request. Data in the request was invalid.”

},

{

“code”: 412,

“type”: 300,

“subCode”: 0,

“message”: “Previous request required. A contact creation request must be made before this one, and was not”

},

{   “code”: 413,

   “subCode”: 0,

   "type":"ValidationException",

    “message”: “Value out of acceptable range. The value ‘%1’ is not supported

for the field ‘%2’ by this service. Valid values must be between

‘%3’ and ‘%4’.

    “substitutionList:[

            {“id”:”%1”,

             “value”:”999”,

            },

            {“id”:”%2”,

             “value”:”CreditScore”,

            },

            { “id”:”%3”,

             “value”:”250”

            },

           { "id":"%4",

             "value":"899"

           }

    ]

}

]

}

### Error codes

See the Error.xsd for all valid error codes. This method is depreciated

### Error types

See the Error.xsd for all valid error types. This method is depreciated.

# Bibliography

W3C. (n.d.). *Key words for use in RFCs to Indicate Requirement Levels [RFC2119].* Retrieved Sept. 8th, 2011, from W3C.