

Configuration Data and Services 4.0

# Document Status

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

This version: **Assembla**.com. Files Tag = CUFX\_4.0\_RFC\_Active

Previous Version: **Assembla**.com. Files Tag = CUFX\_3.3\_RFC\_Archive

# Authors and Change Log

|  |  |  |
| --- | --- | --- |
| Version | Date | Changes |
| 0.0.01 |  | * Initial Draft |
| 0.0.02 |  | * Changed credential group to configuration in some areas where it was left over from a previous spec |
| 0.0.03 |  | * Added name to customDataUse and fixed name of methodDependencyList |
| 0.0.04 |  | * Added service status |
| 3.0 | **03/12/2014** | * Versioning and format with release CUFX 3.0 |
| 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, Added ConfigurationFilter, and ConfigurationMessage. Updated CufxVersion, moved to common, Reference to Configuration and Message Context |

# Overview of Specification

The Configuration Data Model and Services defines what capabilities the CUFX end points have enabled. The configuration services allow another endpoint to read what services an endpoint has allowed. One configuration service may support multiple financial institutions, so the configuration is stored for each financial institution accessible via the CUFX end point.

* **Methods supported**/limitations that may exist in the calls
* **Fields that are not available**
* **Protocols supported (REST-JSON, REST-XML, SOAP) and version currently supported**
* **Order that the services can be orchestrated.**
* **List any extensions to the specifications (custom Data areas)**
* **Language support.**
* **Currency support.**
* **Time zone.**

**Future versions may support**

* Levels of OData operations that are supported.
* Sort order of data coming out of the services (part of oData)
* Restrictions like maximum payload size

# Any know Errors in the document

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

# Table of Contents

[Document Status 1](#_Toc506700764)

[Authors and Change Log 1](#_Toc506700765)

[Overview of Specification 1](#_Toc506700766)

[Any know Errors in the document 2](#_Toc506700767)

[Table of Contents 2](#_Toc506700768)

[Document Conventions 2](#_Toc506700769)

[Release 4.0 Global Update Notes 3](#_Toc506700770)

[Definitions related to the specification 3](#_Toc506700771)

[Data Elements 3](#_Toc506700772)

[Filters used when accessing the Configuration data 3](#_Toc506700773)

[Configuration Data attributes 4](#_Toc506700774)

[Configuration Services 4](#_Toc506700775)

[Overview 4](#_Toc506700776)

[Configuration Resource based create, read, update, delete services 4](#_Toc506700777)

[REST-JSON READ Configuration example 4](#_Toc506700778)

[General Error handling For All Services 6](#_Toc506700779)

[Bibliography 6](#_Toc506700780)

# 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.

# 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.

# Definitions related to the specification

**Configuration**

The parameters in the configuration define the setup of the CUFX endpoint that receives requests from applications. This allows the application to understand where the endpoint services are located, any limitations or customizations the endpoint may have in its implementation of the CUFX specification.

# Data Elements

## Filters used when accessing the Configuration data

Refer to Security Services documentation to understand what may be contained the header and processed by security procedures. When accessing the data include **MessageContext.xsd** so that the service can determine the scope of the request. Refer to recent CUFX messageContext Data and CUFX Security Services for use of MessageContext.xsd.

IMPORTANT: This filter acts differently than other services within the CUFX library. It uses the value in messageContext for fiId to identify what the configurationList should be populated with.

## Configuration Data attributes

All CUFX fields related to a configuration service are defined in **Configuration.xsd.**

# Configuration Services

## Overview

|  |  |
| --- | --- |
| Definition | Collection of services to read the current capabilities of the CUFX services in general such limitations, |
| Overview of Capabilities | Create, read, update and delete a configuration. |
| Dependencies | Security Services, messageContext, party |
| Sample CUFX REST LINK | https://api.dataprovider.com/configuration |
| CUFX SOAP LINK |  |
| CUFX WaDL LINK |  |

## Configuration Resource based create, read, update, delete services

|  |  |
| --- | --- |
| INPUTS | cufx:configurationMessage (which includes)   * cufx:MessageContext * cufx: configurationFilter (for read, update) * cufx: configurationList (for create, update, delete) |
| Outputs | cufx: configurationMessage (which includes)   * cufx:MessageContext * cufx: configurationList |
| Return Values | cufx: configurationMessage (which includes)   * cufx:MessageContext   + statusList |
| Side Effects | Read has no side effects. |
| Dependencies | Security Services for authentication and security. |
| Fields used | Message Headers : See security services  messageContext: See MessageContext.xsd  Filters: See MessageContext.xsd  Attributes: configurationList: See Configuration.xsd |

### REST-JSON READ Configuration example

When no filters are applied, all configuration information is returned.

**Required**: messageContext.

**REQUEST:**

Headers:

**<security related header parameters... see Security Services>**

Accept: application/json

Accept-Charset: utf-8

Accept-Language: en-us (IANA – language codes)(W3C, HTTP Protocols)

Content-type: application/json; charset=utf-8

X-API-Version: >=3.3.0

X-HTTP-Method-Override: GET

**POST h**ttps://api.datasource.com/configurationMessage

{

"configurationMessage": {

"@xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",

"messageContext": {

"cufxVersion": "4.0.0",

"requestId": "requestId1",

"vendorId": "vendorId1",

"appId": "appId1",

"fiId": "My Favorite FI",

"dataSourceId": "dataSourceId1",

"environment": "Development",

"returnDataFilter": "All",

"includeBlankFields": "true",

"includeZeroNumerics": "true"

},

"configurationFilter": {

"fiIdList": {

"fiId": "My Favorite FI"

},

}

}

}

**RESPONSE:**

Headers:

Status Code: 200 Ok

Content-type: application/json; charset=utf-8

Content-Language: en-us

Payload:

{

"configurationMessage": {

"@xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",

"messageContext": {

"cufxVersion": "4.0.0",

"requestId": "requestId1",

"vendorId": "vendorId1",

"appId": "appId1",

"fiId": "My Favorite FI",

"dataSourceId": "dataSourceId1",

"environment": "Development",

"returnDataFilter": "All",

"includeBlankFields": "true",

"includeZeroNumerics": "true",

"statusList": {

"status": {

"statusType": "Success"

},

},

“configurationList”: [

{

“fiIdList”:[{“fiId”:”My Favorite FI”}],

“endPointTimeZOneUTOffset”:-3,

“serviceList”: [

{“serviceName”:”**party**”,

“protocol”:”REST-JSON”,

“cufxVersion”:”2.0.0”,

“currencySupportList: [{“currencySupport”:”USD”}],

“acceptLanguageList”:[{“acceptLanguage:”en-us”}],

“fieldsNotSupportList”:[

{“xsdFile”:”Party.xsd”,

“xsdXPath”:”/partyList/party/characteristics/individual/citizenshipList/\*”,

“comment”:”Back end core does not support this field”

}

],

“serviceTimeZoneUTOffset”:-4,

“status”:”Online”,

“customDataUseList”:[

{“xsdFile”:”Party.xsd”,

“xsdXPath”:”/partyList/party/customData/\*”,

“name”:”eyecolor”,

“comment”:”Will hold the party’s eyecolor in a ValuePair”

}

],

“methodList”:[

{“methodName”:”Create”,

“uri”:”POST https://api.dataprovider.com/party”,

“methodDependencyList”:[

{“serviceName”:”Relationship”,

“methodName”:”Create”,

“cufxVersion”:”2.0.0”}

]

},

{“methodName”:”Read”,

“uri”:”POST https://api.dataprovider.com/deposit?subMethod=GET”

},

{“methodName”:”Update”,

“uri”:”PUT https://api.dataprovider.com/deposit”

},

{“methodName”:”DELETE”,

“uri”:”PUT https://api.dataprovider.com/deposit?subMethod=DELETE”

}

}

]

}

]

}

# General Error handling For All Services

Refer to latest CUFX documentation *Error Mapping*.

Bibliography

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