

Document Data and Services 4.1

# 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\_4.1\_RFC\_Active

Previous Version: **Assembla**.com. Files Tag = CUFX\_4.0\_RFC\_Archive

# Change Log

|  |  |  |
| --- | --- | --- |
| Version | Date | Changes |
| 0.0.01 |  | * Initial Draft |
| 0.0.02 |  | * Update id List for consistency and make sure array’s were defined properly |
| 0.0.03 |  | * Renamed to document services to make it more generic and cover more document types. |
| 3.0 | **10/29/2013** | * Switch to use X-HTTP-METHOD-OVERRIDE standard rather than subMethod non-Standard method for overriding request types. * Create a documentMessage wrapper for every message to increase ability for infrastructure to serialize the data * Versioning and format change with release CUFX 3.0 |
| 3.0 | **12/13/2013** | * Update examples X-API-Version to >=3.0.0 |
| 3.0 | **01/07/2014** | * Updated JSON Examples. |
| 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, Date Range Global Update, Microsoft Global bug fix |
| 4.1 | **12/10/2018** | * Updated to release 4.1, Changed element signedWithIpAddress to use type common:IpAddress |

# Overview of Specification

The Document Data Model and Services defines the information stored when a person signs a document or agreement electronically or manually such as loan document, membership agreements, etc. The model and services also relate the document to the party(ies), relationship and/or account(s)s to which it relates. Documents can be created, read, updated or deleted within the back end system via the service methods. In addition, to storing final documents, document services also contains the concept of a template documents for selection by the document type, sub type and the scenario to which it applies (such as particular state, SEG or product). Document templates can be pulled up with the member information pre-filled by requesting that it be prefilled in the service request instructions. The actual image (s) of the document is stored using artifact services.

Documents and document templates can be effective dated and expired if necessary.

# Any know Errors in the document

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

# Table of Contents

[Document Status 1](#_Toc532213756)

[Change Log 1](#_Toc532213757)

[Overview of Specification 1](#_Toc532213758)

[Any know Errors in the document 2](#_Toc532213759)

[Table of Contents 2](#_Toc532213760)

[Document Conventions 2](#_Toc532213761)

[Release 4.0 Global Update Notes 3](#_Toc532213762)

[Definitions related to the specification 3](#_Toc532213763)

[Data Elements 3](#_Toc532213764)

[Filters used when accessing the Document data 3](#_Toc532213765)

[Document Data attributes 4](#_Toc532213766)

[Document Services 4](#_Toc532213767)

[Overview 4](#_Toc532213768)

[Document Resource based create, read, update, delete services 4](#_Toc532213769)

[REST-JSON READ Document Template example 5](#_Toc532213770)

[REST-JSON CREATE Document example 6](#_Toc532213771)

[REST-JSON READ Document example 7](#_Toc532213772)

[REST-JSON UPDATED Document example 8](#_Toc532213773)

[REST-JSON DELETE Document example 9](#_Toc532213774)

[General Error handling For All Services 10](#_Toc532213775)

[Bibliography 10](#_Toc532213776)

# 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

**Document**

Textual and graphical information that conveys information that is either information or an agreement between legal entities.

**Disclosure**

A document, notification, web page, etc. that has legal information that is required to be agreed to by the party(ies) related to accounts or a relationship or just sent to or viewed by the party(ies).

# Data Elements

## Filters used when accessing the Document 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.

Include any filter variables related to the request. See **DocumentFilter.xsd.**

## Document Data attributes

All CUFX fields related to the document service are defined in **Document.xsd.**

# Document Services

## Overview

|  |  |
| --- | --- |
| Definition | Collection of services to create, read, update or delete the documents or document templates within the backend system. The document data defines the information stored when a person signs a document or agreement electronically or manually. The document data relates the documents to the party, relationship and/or account to which it relates. Document templates can be pulled up and prefilled with party, account or relationship data. |
| Overview of Capabilities | Create, read, update and delete a document, documents, or document templates. |
| Dependencies | Security Services, messageContext, party |
| Sample CUFX REST LINK | https://api.dataprovider.com/document |
| CUFX SOAP LINK |  |
| CUFX WaDL LINK |  |

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

|  |  |
| --- | --- |
| INPUTS | cufx:documentMessage (which includes)   * cufx:messageContext * cufx:documentFilter (read, update) * cufx:documentList (create, update, delete) |
| Outputs | cufx:documentMessage (which includes)   * cufx:messageContext * cufx:documentList |
| Return Values | cufx:documentMessage (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 DocumentFilter.xsd  Attributes: documentList: See Document.xsd |

### REST-JSON READ Document Template example

This example shows a Truth in Lending document template being retrieved.

**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-HTTP-Method-Override: GET**

X-API-Version: >=4.0.0

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

{

“documentMessage”: {

“messageContext”: { <see MessageContext.xsd>

},

"documentList": [{

“document” :[{

“prefillable”:”False”

}]

}],

“documentFilter”: {

“documentTypeList”:[”Disclosure”],

“documentSubTypeList”:[“TrutheInLending”],

“filterDateTime”: ”2013-03-01T05:21Z”,

“documentStatusList”:[“Template”]

}

}

}

**RESPONSE:**

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

“documentMessage”: {

“messageContext”: { <see MessageContext.xsd>

},

“documentList”: [{

“document” : [{

“documentId”:”848182”,

“documentTitle”:”Truth In Lending”,

“documentType”:”Disclosure”,

“documentSubType”:”TruthInLending”,

“documentStatus”:”Template”,

“documentVersion”:”1.0”,

“effectiveDateTime”:”2011-01-01T00:00Z”,

“expirationDateTime”:”2014-12-31T00:00Z”,

“documentArtifactIdList”:{

“artifactId”:{“artifactUniqueId”:”521339854”},

},

“prefillable”:”False”,

“prefilled”:”False”,

}]

}]

}

}

### REST-JSON CREATE Document example

This example shows taking the previous template and creating a disclosure document for a two parties on an account.

**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: >=4.0.0

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

{

“documentMessage”: {

“messageContext”: { <see MessageContext.xsd>

},

“documentList”: [{

“document” : [{

“documentTitle”:”Truth In Lending”,

“documentType”:”Disclosure”,

“documentSubType”:”TruthInLending”,

“documentStatus”:”Sent”,

“documentVersion”:”1.0”,

“effectiveDateTime”:”2013-03-28T12:35Z”,

“expirationDateTime”:”2014-03-28T12:35Z”,

“documentArtifactIdList”:{

“artifactId”:{“artifactUniqueId”:”153415534h34h53a”},

},

“partyIdList”:[”13534”,”3534135534”],

“accountIdList”:[”12315”],

“prefillable”:”False”,

“prefilled”:”False”,

}]

}]

}

}

**RESPONSE:**

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

“documentMessage”: {

“messageContext”: { <see MessageContext.xsd>

},

“documentList”: [{

“document” : [{

“documentId”:”5451542871asd54e31”,

“documentTitle”:”Truth In Lending”,

“documentType”:”Disclosure”,

“documentSubType”:”TruthInLending”,

“documentStatus”:”Sent”,

“documentVersion”:”1.0”,

“effectiveDateTime”:”2013-03-28T12:35Z”,

“expirationDateTime”:”2014-03-28T12:35Z”,

“documentArtifactIdList”:{

“artifactId”:{“artifactUniqueId”:”153415534h34h53a”},

},

“partyIdList”:[”13534”,”3534135534”],

“accountIdList”:[”12315”],

“prefillable”:”False”

“prefilled”:”False”,

}]

}]

}

}

### REST-JSON READ Document example

This example shows a document being read for a two parties on an account.

**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-HTTP-Method-Override: GET**

X-API-Version: >=4.0.0

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

{

“documentMessage”: {

“messageContext”: { <see MessageContext.xsd>

},

“documentFilter”: {

“documentIdList”:[”5451542871asd54e31”]

}

}

}

**RESPONSE:**

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

“documentMessage”: {

“messageContext”: { <see MessageContext.xsd>

},

“documentList”: [{

“document” : [{

“documentId”:”5451542871asd54e31”,

“documentTitle”:”Truth In Lending”,

“documentType”:”Disclosure”,

“documentSubType”:”TruthInLending”,

“documentStatus”:”Sent”,

“documentVersion”:”1.0”,

“effectiveDateTime”:”2013-03-28T12:35Z”,

“expirationDateTime”:”2014-03-28T12:35Z”,

“documentArtifactIdList”:{

“artifactId”:{“artifactUniqueId”:”153415534h34h53a”},

},

“partyIdList”:[”13534”,”3534135534”],

“accountIdList”:[”12315”],

“prefillable”:”False”,

“prefilled”:”False”,

}]

}]

}

}

### REST-JSON UPDATED Document example

This example shows a document being updated for a two parties on an account.

**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: >=4.0.0

**PUT h**ttps://api.datasource.com/document

{

“documentMessage”: {

“messageContext”: { <see MessageContext.xsd>

},

“documentFilter”: {

“documentIdList”:[”5451542871asd54e31”]

},

“documentList”: [{

“document” : [{

“documentId”:”5451542871asd54e31”,

“documentStatus”:”Received”,

“documentSignatureList”:[{

“documentSignatureType”:”SingleClickAcceptance”,

“documentSignedDate”:“2013-01-28T11:03Z”,

“signaturePartyid”:”3534135534”,

“signedWithIpAddress”:”111.34.35.123”,

“documentDigitalFingerprint”:”1235FF1AE1235”,

“documentDigitalAlgorithm”:“SHA-1”,

“documentDigitalCertificate”:”11519987198118aae993fg991j3…”,

“usersPublicKey”:”135876a8167343…”

}]

}]

}

}

**RESPONSE:**

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

“documentMessage”: {

“messageContext”: { <see MessageContext.xsd>

},

“documentList”: [{

“document” : [{

“documentId”:”5451542871asd54e31”,

“documentTitle”:”Truth In Lending”,

“documentType”:”Disclosure”,

“documentType”:”TruthInLending”,

“documentStatus”:”Received”,

“documentVersion”:”1.0”,

“effectiveDateTime”:”2013-03-28T12:35Z”,

“expirationDateTime”:”2014-03-28T12:35Z”,

“documentArtifactIdList”:{

“artifactId”:{“artifactUniqueId”:”153415534h34h53a”},

},

“documentSignatureType”:”WetSignature”

“documentSignedDate”:“2013-01-28T11:03Z”,

“partyIdList”:[”13534**”,”3534135534”**],

“accountIdList”:[”12315”],

“prefillable”:”False”,

“prefilled”:”False”,

“documentSignatureList”:[{

“documentSignatureType”:”SingleClickAcceptance”,

“documentSignedDate”:“2013-01-28T11:03Z”,

“signaturePartyid”:”3534135534”,

“signedWithIpAddress”:”11.34.35.123”,

“documentDigitalFingerprint”:”1235FF1AE1235”,

“documentDigitalAlgorithm”:“SHA-1”,

“documentDigitalCertificate”:”11519987198118aae993fg991j3…”,

“usersPublicKey”:”135876a8167343…”

]]

}]

}

}

### REST-JSON DELETE Document example

This example shows a document being deleted for a two parties on an account.

**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-HTTP-Method-Override: DELETE**

X-API-Version: >=4.0.0

**PUT h**ttps://api.datasource.com/document

{

“documentMessage”: {

“messageContext”: { <see MessageContext.xsd>

},

“documentFilter”: {

“documentIdList”:[”5451542871asd54e31”]

}

}

}

**RESPONSE:**

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

"documentMessage": {

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

"messageContext": {

"cufxVersion": "1.0.3",

"requestId": "requestId1",

"vendorId": "vendorId1",

"appId": "appId1",

"fiId": "fiId1",

"dataSourceId": "dataSourceId1",

"environment": "Development",

"returnDataFilter": "All",

"includeBlankFields": "true",

"includeZeroNumerics": "true",

"user": {

"userId": "userId1",

"processorSessionId": "processorSessionId1",

"userType": "EmployeeId"

},

"statusList": {

"status": {

"statusType": "Success"

}

}

}

}

}

# 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.