

Secure Message 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 |
| 2.1.0 |  | * Initial Creation |
| 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 secureMessageMessage wrapper for every message to increase ability for infrastructure to serialize the data * Versioning and format change with release CUFX 3.0 |
| 3.0 | **12/16/2013** | * Update examples X-API-Version to >=3.0.0 |
| 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, Date Range Global Update, Microsoft Global bug fix |
| 4.1 | **12/10/2018** | * Updated to release 4.1, * Replaced messageContext reference with User. Removed partId from SecureMessageUser as partyId is now included in the User extension base. Added allowReply boolean |

# Overview of Specification

The CUFX secure message data and services is a collection of services that allows a client application to communicate with a back end repository of secure messages. This model is flexible in that a secure message may be related to a sending/receiving fiUserId, to multiple parties, accounts, relationships, or cards. The secure message may have multiple documents attached to it (see CUFX document data and services) which, if the document service supports it, may be electronically signed.

# Known Errors in the document

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

# Table of Contents

Document Status 1

Change Log 1

Overview of Specification 1

Known Errors in the document 2

Table of Contents 2

Document Conventions 2

Release 4.0 Global Update Notes 3

Definitions related to the specification 3

Data Elements 3

DATA ELEMENT: SecureMessageMessage 4

DATA ELEMENT: SecureMessage 4

DATA ELEMENT: SecureMessageList 5

DATA ELEMENT: SecureMessageUser 5

DATA ELEMENT: SecureMessageUserList 5

DATA ELEMENT: SecureMessageStatus 5

DATA ELEMENT: SecureMessageType 5

Filters used when accessing the SecureMessage data 6

DATA ELEMENT: SecureMessageFilter 6

SecureMessage Data attributes 7

Services 7

Overview 7

SecureMessage Resource based create, read, update, delete services 7

REST-JSON READ SECUREMESSSAGE list of available SecureMessage templates 8

REST-JSON CREATE SECUREMESSSAGE 9

REST-JSON READ SECUREMESSSAGE of a specific secureMessage ID example 10

REST-JSON UPDATE SECUREMESSSAGE for with specific SecureMessage ID. 10

REST-JSON DELETE SECUREMESSSAGE example. 11

General Error handling For All Services 12

Bibliography 12

# 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

secureMessage

SecureMessage is a collection of attributes that represent a message to which only the intended receiver should have access. The secure message may have attached documents. Those documents may be electronically signed (See CUFX Document Data and Services).

RECEIVER

A receiver may be either a person identified by a fiUserId, party identifier, set of accounts, relationship or cards.

# Data Elements

The important complex, container, and enumeration elements making up the CUFX SecureMessage service are described below. The full element specifications may be found in **SecureMessage.xsd, SecureMessageMessage.xsd**, and **SecureMessageFilter.xsd.** Common elements can be found in **Common.xsd**.

### DATA ELEMENT: SecureMessageMessage

The SecureMessageMessage object is a wrapper object that contains a SecureMessage for transmission.

**Attributes**

* ***messageContext*** – The CUFX MessageContext for this object. Note that this is unrelated to the SecureMessage service specifically, and is related to the CUFX request/response model. The MessageContext object identifies the sender and provides some level of security or processing information for any given CUFX request.
* ***secureMessageFilter*** – Any SecureMessageFilter object being transmitted in this SecureMessageMessage.
* ***secureMessageList*** – Any SecureMessageList object (*i.e.*, list of zero or more SecureMessage objects) being transmitted in this SecureMessageMessage.

### DATA ELEMENT: SecureMessage

The SecureMessage object defines the information sent and then received by two or more parties. The object also relates the message to the sending and receiving party or parties, potentially through indicating the relationship of these parties to account(s) or other data elements.

**Attributes**

* ***secureMessageId*** – A persistent unique identifier for the object, not related to any entities references in the object.
* ***previousSecureMessageId*** – The secureMessageId of any immediately preceding message in a series containing this secureMessage object.
* ***threadId*** – The identifier of the thread containing this secureMessage
* ***subjectLine*** - The text subject of the message.
* ***messageFromList*** – A SecureMessageUserList object containing the user(s) originating the message. Typically, this list will contain only a single SecureMessageUser object.
* ***messageToList*** - A SecureMessageUserList object containing the user(s) to receive the message.
* ***carbonCopyList*** - A SecureMessageUserList object containing the user(s) to receive the message as a Cc:.
* ***blindCarbonCopyList*** - A SecureMessageUserList object containing the user(s) to receive the message as a Bcc:.
* ***replyToList*** - A SecureMessageUserList object containing the user(s) to receive any reply to this secureMessage.
* ***type*** - The SecureMessageType of this SecureMessage object.
* ***currentStatus*** – The current SecureMessageStatus of this secureMessage.
* ***statusLog*** – A StatusLog object indicating dates and times when the secureMessage was altered or transmitted.
* ***body*** - Contains the text of the secureMessage.
* ***bodyFormat*** - The text format of the body (e.g. HTML, RTF, Plain Text).
* ***documentIdList*** – A list containing the identifiers for the zero or more documents attached to the secureMessage.
* ***doNotDeliverBeforeDateTime*** – An optional time before which the secureMessage should not be delivered to its recipients.
* ***expirationDateTime*** – An optional time when the message expires and should be left undelivered or deleted if not viewed (or handled in any other specific implementation-dependent way).
* ***requestId*** – An optional ProductServiceRequest object identifier indicating a specific product service request to which this secureMessage relates.
* ***customDate*** – Implementation-specific key/value pairs to be used as a last resort for customization of behavior or data content.

### DATA ELEMENT: SecureMessageList

The SecureMessageList object is a simple collection object containing zero or more SecureMessage objects.

### DATA ELEMENT: SecureMessageUser

The SecureMessageUser object identifies one or more persons who have roles relative to a SecureMessage object. Any one or more of the attributes in the element may be populated and all are considered to identify a set of persons.

**Attributes**

* ***partyId*** – A CUFX PartyId identifying a person.
* ***relationshipId*** – A CUFX relationshipId identifying a relationship; the person(s) party to this relationship are therefore the person(s) identified by this SecureMessageUser object.
* ***accountId*** – A CUFX AccountId identifying an account; person(s) owning or having particular rights on this account are therefore the person(s) identified by this SecureMessageUser object.
* ***cardId*** – A CUFX CardId identifying an card; person(s) owning or having particular rights on this card are therefore the person(s) identified by this SecureMessageUser object.

### DATA ELEMENT: SecureMessageUserList

The SecureMessageUserList object is a simple collection object containing zero or more SecureMessageUser objects.

### DATA ELEMENT: SecureMessageStatus

The SecureMessageStatus object is an enumeration containing the following values:

* Draft
* Outbox
* Sent
* Read
* Replied
* Forwarded
* Archived
* Unread
* Failed
* Expired

### DATA ELEMENT: SecureMessageType

The SecureMessageType object is an enumeration containing the following values:

* Alert
* GeneralMessage
* Notice
* ProductOffering
* ScheduleMeeting
* ServiceUpdate
* SignatureRequired
* Other

## Filters used when accessing the SecureMessage 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 **SecureMessageFilter.xsd.**

The filters are used to filter based on associated data type for the secureMessage such as secureMessages for an account, party, contact, etc. Combining filters can be used to get secureMessages for such as things as Alert secureMessages on a specific account or associated to a specific party or contact, web site secureMessages for the logged in party, etc.

### DATA ELEMENT: SecureMessageFilter

The SecureMessageFilter object is used to identify zero or more SecureMessage objects that match the criteria contained by the filter. This object is described in **SecureMessageFilter.xsd** and contains attributes allowing filtering of messages by any of the attributes of the SecureMessage object.

**Attributes**

* ***secureMessageIdList*** – A list containing zero or more identifiers of particular SecureMessage objects.
* ***threadIdList*** – A list containing zero or more identifiers of particular Thread objects.
* ***subjectLineContainsList*** – A list containing zero or more strings that will be contained in the subjectLine attributes of all matching SecureMessage object(s).
* ***messageFromList*** – A SecureMessageUserList object identifying the SecureMessageUser(s) that are among the senders of the matching SecureMessage object(s).
* ***messageToList*** – A SecureMessageUserList object identifying the SecureMessageUser(s) that are among the recipients of the matching SecureMessage object(s).
* ***carbonCopyList*** – A SecureMessageUserList object identifying the SecureMessageUser(s) that are among the Cc: recipients of the matching SecureMessage object(s).
* ***blindCarbonCopyList*** – A SecureMessageUserList object identifying the SecureMessageUser(s) that are among the Bcc: recipients of the matching SecureMessage object(s).
* ***replyTo*** – A SecureMessageUserList object identifying the SecureMessageUser(s) that are held in the replyTo attributes of the matching SecureMessage object(s).
* ***secureMessageTypeList*** – A SecureMessageTypeList object identifying the SecureMessageType(s) of the matching SecureMessage object(s).
* ***currentStatusList*** – A SecureMessageStatusList object identifying the SecureMessageStatus(es) of the matching SecureMessage object(s).
* ***statusLogList*** – A SecureMessageStatusList object identifying the SecureMessageStatus(es) contained (in the statusLog of each) as the current or past statuses of the matching SecureMessage object(s).
* ***statusLogBeginDate*** – An optional earliest date and time to be used when filtering using a provided statusLogList attribute (described above).
* ***statusLogEndDate*** – An optional latest date and time to be used when filtering using a provided statusLogList attribute (described above).
* ***bodyContainsList*** – A list containing zero or more strings that will be contained in the body attributes of all matching SecureMessage object(s).
* ***returnConversation*** – A true or false attribute used to indicate whether the response should contain all previous messages linked to those identified by the filters in other attributes.
* ***documentIdList*** – A list containing zero or more documentId's specifying the CUFX Document objects that are attached to the SecureMessages being identified.
* ***productServiceRequestIdlist*** – A list containing zero or more ProductServiceRequest identifiers specifying the product service requests related to the SecureMessages being identified.

## SecureMessage Data attributes

All CUFX fields related to a secureMessage are defined in **SecureMessage.xsd**.

# Services

## Overview

|  |  |
| --- | --- |
| Definition | Collection of services to manage a secureMessage |
| Overview of Capabilities | Create, read, update and delete a secureMessage. The following scenarios may exist. The secureMessage may be connected to an existing relationship, account, party or card. |
| Dependencies | fiUserId, Account, Party, card, and relationship data and security services |
| Sample CUFX REST LINK | https://api.dataprovider.com/secureMessage |
| CUFX WaDL LINK |  |

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

|  |  |
| --- | --- |
| INPUTS | cufx: secureMessageMessage (which includes)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * cufx:secureMessageFilter (for read, update, delete) * cufx:secureMessage (for create, update) |
| Outputs | cufx: secureMessageMessage (which includes)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * cufx:secureMessage |
| Return Values | cufx: secureMessageMessage (which includes)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html)   + statusList |
| Side Effects | Creation, update or deletion of secureMessage. 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 SecureMessageFilter.xsd  Attributes: secureMessage : See SecureMessage.xsd  Errors: See MessageContext.xsd |

### REST-JSON READ SECUREMESSSAGE list of available SecureMessage templates

This example pulls all available template records so system knows what available secure messages a party can use.

Note: Not all fields are listed for simplicity of an example to create a secureMessage.

**Required**: messageContext, **at least one filter in secureMessageFilter.**

**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 https://api.datasource.com/secureMessage

{

"secureMessageMessage": {

"messageContext": { <see MessageContext.xsd>

},

"**secureMessageFilter**":{

"**secureMessageStatusList**":[ "Draft" ],

}

}

}

**RESPONSE**:

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

"secureMessageMessage": {

"messageContext": { <see MessageContext.xsd>

},

"secureMessageList":[

{"**secureMessageId**":"12334",

"subjectLine":"Credit Card Dispute",

"messageToList":[

{"partyId":"254561278f112"}

],

"type":"ServiceUpdate",

"currentStatus":"Draft",

"body":"Credit Card Number<br />Transaction Description<br />",

"bodyFormat":"Html",

"secureMessageStatus":"Draft"

},

*<other matching messages>*

…

]

}

}

### REST-JSON CREATE SECUREMESSSAGE

This example shows where the relationship, party and contact are known.

Note: Not all fields are listed for simplicity of an example to create a secureMessage.

**Required**: messageContext, at least one secureMessage.

**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/secureMessage

{

"secureMessageMessage": {

"messageContext": { <see MessageContext.xsd>

},

"secureMessageList":[

{"messageToList":[

{"partyId":"254561278f112"},

{"partyId":"25661237612a3"},

{"accountId":"15642123"}

],

"type":"GeneralMessage",

"subjectLine":"New message from your Credit Union!",

"body":"Your Credit Union proudly uses CUFX!",

"secureMessageStatus":"Draft",

}

]

}

}

RESPONSE:

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

"secureMessageMessage": {

"messageContext": { <see MessageContext.xsd>

},

"secureMessageList":[

{"**secureMessageId**":"7070615644aasdfasdfa535478932sf",

"messageToList":[

{"partyId":"254561278f112"},

{"partyId":"25661237612a3"},

{"accountId":"15642123"}

],

"type":"GeneralMessage",

"subjectLine":"New message from your Credit Union!",

"body":"Your Credit Union proudly uses CUFX!",

"secureMessageStatus":"Draft",

}

]

}

}

### REST-JSON READ SECUREMESSSAGE of a specific secureMessage ID example

This example shows where the secureMessage ID is known.

Note: Not all fields are listed for simplicity of an example to create a secureMessage.

**Required**: messageContext, **at least one filter in secureMessageFilter.**

**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 https://api.datasource.com/secureMessage

{

"secureMessageMessage": {

"messageContext": { <see MessageContext.xsd>

},

"**secureMessageFilter**":{

"**secureMessageIdList**":[ "7070615644aasdfasdfa535478932sf" ]

}

}

}

**RESPONSE**:

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

"secureMessageMessage": {

"messageContext": { <see MessageContext.xsd>

},

"secureMessageList":[

{"**secureMessageId**":"7070615644aasdfasdfa535478932sf",

"messageToList":[

{"partyId":"15353534155asdf5"}

],

"type":"GeneralMessage",

"subjectLine":"New message from your Credit Union!",

"body":"Your Credit Union proudly uses CUFX!"

}

]

}

}

### REST-JSON UPDATE SECUREMESSSAGE for with specific SecureMessage ID.

This example shows where the secureMessage ID’s are known.

Note: Not all fields are listed for simplicity of an example to create a secureMessage.

**Required**: messageContext, at least one filter in secureMessageFilter.

**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 https://api.datasource.com/secureMessage

{

"secureMessageMessage": {

"messageContext": { <see MessageContext.xsd>

},

"**secureMessageFilter**":{

"**secureMessageIdList**": [ "7070615644aasdfasdfa535478932sf" ]

}

"secureMessageList":[

{"secureMessageStatus":"Archived"

}

]

}

}

**RESPONSE**:

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

"secureMessageMessage": {

"messageContext": { <see MessageContext.xsd>

},

"secureMessageList":[

{"secureMessageId":"7070615644aasdfasdfa535478932sf",

"messageToList": [ { "partyId": "15353534155asdf5" } ],

"type":"GeneralMessage",

"secureMessageStatus":"Archived",

"subjectLine":"New message from your Credit Union!",

"body":"Your Credit Union proudly uses CUFX!"

}

]

}

}

### REST-JSON DELETE SECUREMESSSAGE example.

This example deletes all secureMessage records from a list of account Id’s.

Note: Not all fields are listed for simplicity of an example to create a secureMessage.

**Required**: messageContext, at least one filter in secureMessageFilter.

**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 https://api.datasource.com/secureMessage

{

"secureMessageMessage": {

"messageContext": { <see MessageContext.xsd>

},

"**secureMessageFilter**":{

"accountIdList":[

615644aa32sf",

"15345151s25",

"1513341635"

]

]

}

}

**RESPONSE**:

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

"secureMessageMessage": {

"messageContext": {

"cufxVersion": "4.0.0",

"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": {

"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.