

Remote Deposit Capture Model and Services 4.4

Document Status

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

Change Log

|  |  |  |
| --- | --- | --- |
| Version | Date | Changes |
| 0.0.1 |  | * Initial Creation |
| 0.0.2 |  | * Updates and added examples |
| 3.0 | **10/29/2013** | * Create a remoteDepositRequestMessage or a remoteDepositPostRequestMessage 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 |
| 4.2 | **03/05/2019** | * Updated to release 4.2, \*\*\* 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. * RemoteDeposit - added enums AmountMismatch and missing to type ImageValidationIssue. |
| 4.3 | **10/07/2019** | * Updated to release 4.3, renamed file removing version as proper version control is being used in Github. Added FeeSchedule to namespace. Updated documentation for “amount”. Added elements userEnteredAmount, immediatelyAvailableAmount, remoteDepositFeeList, businessDaysHeld. Added enums to RemoteDepositStatus of values: UnacceptableRisk, UnknownUser, InvalidAccount, and ConfirmationRequired. Added element imageValidationIssueOther. In ImageValidationIssue, deprecated CARMismatchFailed and added ImageQuality, Unknown, Other. Corrected documentation spelling errors in RemoteDepositActionList and RemoteDepositIdList. |
| 4.4 | **10/20/2020** | * Updated to release 4.4, RemoteDeposit - Removed transaction namespace reference. Updated checkNumber to common type reference. * Added accountType and accountSubType to RemoteDeposit * RemoteDepositFilter - Added accountIdentificationList to filter |

Overview of Specification

The Remote Deposit Capture specification describes the services used as part of an offering to allow a user to remotely deposit checks. A typical example is mobile remote deposit capture, where a user will use their device (e.g. iPhone) take a picture of a check, which would then deposit the funds into the users account if accepted. This specification is not limited to mobile remote deposit capture, and could also be used for non-mobile devices as well (e.g. desktop remote deposit capture).

This specification describes the data structures and services for remote deposit capture.

Any know Errors in the document

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

Table of Contents

[Document Conventions 3](#_Toc54094050)

[CUFX API and Documentation Support 3](#_Toc54094051)

[Release 4.0 Global Update Notes 3](#_Toc54094052)

[Release 4.4 Global Update Notes 4](#_Toc54094053)

[High level use cases 4](#_Toc54094054)

[Use Case 1: remotely deposit a check 4](#_Toc54094055)

[Use Case: Submit One or More Remote DEposit Requests 4](#_Toc54094056)

[Data Elements 5](#_Toc54094057)

[Remote DEposit Capture Data attributes 5](#_Toc54094058)

[Data Element: RemoteDepositRequestList 5](#_Toc54094059)

[Data Element: RemoteDepositRequest 5](#_Toc54094060)

[Data Element: RemoteDepositResponseList 5](#_Toc54094061)

[Data Element: RemoteDepositResponse 5](#_Toc54094062)

[Data Element: RemoteDepositPostRequestList 6](#_Toc54094063)

[Data Element: RemoteDepositPostRequest 6](#_Toc54094064)

[Data Element: MessageContext 6](#_Toc54094065)

[Remote Deposit Capture Services 6](#_Toc54094066)

[Service Definitions 6](#_Toc54094067)

[Service Message: Remote Deposit 6](#_Toc54094068)

[Service Message: Remote Deposit Post Requests 8](#_Toc54094069)

[Bibliography 11](#_Toc54094070)

# 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.

# High level use cases

## Use Case 1: remotely deposit a check

This use case assumes that the calling application has already:

* Authenticated the user in some manner. Assuming that this is within an online banking system, the online banking system would be responsible for authenticating the user.
* The user is already eligible for the remote deposit capture service. We are assuming that the online banking system would have denied the user to this service already if they were not eligible.
* If the third party remote deposit capture service required an initial registration of the user if this is their first time using the service, that this would have already taken place.

## Use Case: Submit One or More Remote DEposit Requests

CUFX Application

1

SubmitRemoteDepositRequest

Contains list of RemoteDepositRequest elements

2

SubmitRemoteDeposit Response

Contains list of RemoteDepositResponse elements

CUFX Remote Deposit Data Provider

3

RemoteDepositPostRequest Request

Contains list of RemoteDepositPost elements

4

RemoteDepositPostResponse Response

Contains list of RemoteDepositPostResponse elements

This use case shows the remote deposit request being made from the CUFX Application (such as the Online Banking system) to the CUFX Remote Deposit Data Provider (such as Vertifi, Bluepoint, etc). Steps 3 and 4 show the remote deposit capture provider submitting a post request to the CUFX application, which not all remote deposit capture providers work in this way or support this. In some cases the remote deposit capture provider will generate an x.9 file and deliver this to your Check21 system for processing.

# Data Elements

## Remote DEposit Capture Data attributes

The following CUFX fields referenced in the services defined below are utilized for remote deposit capture:

### Data Element: RemoteDepositRequestList

A collection or list of one or more RemoteDepositRequest elements. This specification allows for multiple remote deposit capture items to be submitted in one call.

### Data Element: RemoteDepositRequest

The RemoteDepositRequest data element contains all the attributes to submit a remote deposit to a CUFX remote deposit provider (such as Vertifi, BluePoint, etc) for a decision on whether the deposit will be approved and eventually the deposit made into the members account.

### Data Element: RemoteDepositResponseList

A collection or list of one or more RemoteDepositResponse elements. This specification allows for multiple remote deposit capture items to be submitted in one call. There should be a response for each remote deposit request item in the original request.

### Data Element: RemoteDepositResponse

The RemoteDepositResponse data element contains attributes pertaining to a request to remotely deposit an item. This includes information such as the status of the remotely deposited item (such as Approved, Under Review, Denied), as well as information on why an item was reject, such as poor image quality, over deposit limit, etc.

### Data Element: RemoteDepositPostRequestList

A collection or list of one or more RemoteDepositPostRequest elements. This specification allows for multiple remote deposit capture items to be posted in one call.

### Data Element: RemoteDepositPostRequest

The RemoteDepositPost data element contains all the attributes for posting the funds to an account for a previously remotely deposited item that was approved. This includes attributes such as the posting amount.

### Data Element: MessageContext

Like many of the other CUFX service, a MessageContext object must be passed for each request. Refer to the Security Services documentation for more information.

# Remote Deposit Capture Services

## Service Definitions

The remote deposit capture services support a Request/Response model for submit remote deposit capture requests.

### Service Message: Remote Deposit

One or more remote deposit request elements can be submitted at a time using the RemoteDeposit service.

|  |  |
| --- | --- |
| INPUTS | cufx:remoteDepositMessage (which includes)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * [cufx:RemoteDeposit](file:///\\files2\users\CMarjaniemi\Projects\CUFX\Generated%20HTML%20Docs\Loan.html)Filter (for read, update) * [cufx:RemoteDepositList](file:///\\files2\users\CMarjaniemi\Projects\CUFX\Generated%20HTML%20Docs\Loan.html) (for create, update, delete) |
| Outputs | cufx:remoteDepositMessage (which includes)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * [cufx:RemoteDepositList](file:///\\files2\users\CMarjaniemi\Projects\CUFX\Generated%20HTML%20Docs\Loan.html) |
| Return Values | ufx:remoteDepositMessage (which includes)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html)   + statusList |
| Side Effects | Remotely deposited items posted to the vendor for approval or review. |
| Dependencies | None. |
| CUFX REST LINK | https://api.dataprovider.com/remotedepositMessage |

REST-JSON Example

The following example illustrates how to submit two remote deposit requests.

**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: >=4.3.0

**POST h**ttps://api.dataprovider.com/remotedepositmessage

{

"remoteDepositMessage": {

"messageContext" : <SeeMessageContext.xsd>,

"remoteDepositRequestList" : {

"remoteDepositPostRequest": [

{

"routingTransitNumber": "123345",

"amount": {

"value": 12.34,

"currencyCode": 145

},

"accountId": "1234",

"reference": "ref1234",

"contact": <...See contact XSD...>,

"mode": “Production”

},

{

"routingTransitNumber": "23453",

"amount": {

"value": 72.34,

"currencyCode": 145

},

"accountId": "1234",

"reference": "ref1235",

"contact": <...See contact XSD...>,

"mode": “Production”

}

]

}

}

**RESPONSE:**

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

"remoteDepositMessage": {

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

"messageContext": {

"cufxVersion": "4.3.0",

"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": {

"status": {

"statusType": "Success"

}

},

}

},

"remoteDepositList": {

"currentPage": "1",

"totalPages": "1",

"pageSize": "1",

"positionIndex": "positionIndex1",

"returnPage": "1",

"returnPositionIndex": "returnPositionIndex1",

"remoteDeposit": {

}

}

}

}

### Service Message: Remote Deposit Post Requests

Once a remotely deposit item is accepted, the Remote Deposit Post Request could be used to request that the funds now be remotely deposited to the members account.

One or more remote deposit request elements can be submitted at a time using the RemoteDeposit service.

|  |  |
| --- | --- |
| INPUTS | cufx:remoteDepositMessage (which includes)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * [cufx:RemoteDeposit](file:///\\files2\users\CMarjaniemi\Projects\CUFX\Generated%20HTML%20Docs\Loan.html)Filter (for read, update) * [cufx:RemoteDepositList](file:///\\files2\users\CMarjaniemi\Projects\CUFX\Generated%20HTML%20Docs\Loan.html) (for create, update, delete) |
| Outputs | cufx:remoteDepositMessage (which includes)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * [cufx:RemoteDepositList](file:///\\files2\users\CMarjaniemi\Projects\CUFX\Generated%20HTML%20Docs\Loan.html) |
| Return Values | cufx:remoteDepositMessage (which includes)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html)   + statusList |
| Side Effects | The funds will be posted into the member’s account, usually by taking it from a General Ledger account. |
| Dependencies | None. |
| CUFX REST LINK | https://api.dataprovider.com/remotedepositmessage |

REST-JSON Example

The following example illustrates how to submit two remote deposit post requests.

**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: >=4.3.0

**POST h**ttps://api.dataprovider.com/remotedepositmessage

{

"remoteDepositMessage": {

"messageContext" : <SeeMessageContext.xsd>,

"remoteDepositPostRequest": [

{

"routingTransitNumber": "123345",

"amount": {

"value": 12.34,

"currencyCode": 145

},

"accountId": "1234",

"reference": " ref1234",

"mode": “Production”

},

{

"routingTransitNumber": "routingTransitNumber1",

"amount": {

"value": 1.0,

"currencyCode": 145

},

"accountId": "1234",

"reference": " ref1235",

"mode": “Production”

}

]

}

}

RESPONSE:

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

"remoteDepositMessage": {

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

"messageContext": {

"cufxVersion": "4.3.0",

"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": {

"status": {

"statusType": "Success"

}

},

}

},

"remoteDepositList": {

"currentPage": "1",

"totalPages": "1",

"pageSize": "1",

"positionIndex": "positionIndex1",

"returnPage": "1",

"returnPositionIndex": "returnPositionIndex1",

"remoteDeposit": {

"action": "CreateSession",

"remoteDepositId": "remoteDepositId1",

"clientReferenceId": "clientReferenceId1",

"vendorReferenceId": "vendorReferenceId1",

"checkNumber": "checkNumber1",

"routingTransitNumber": "routingTransitNumber1",

"micrCheckAccountNumber": "micrCheckAccountNumber1",

"amount": {

"value": "1",

"currencyCode": "USD",

"exchangeRate": "1"

},

"cropped": "true",

"frontImage": {

"artifactId": {

"artifactUniqueId": "artifactUniqueId1"

},

"artifactType": "artifactType1",

"externalId": "externalId1",

"artifact": "base64Binary Content",

"artifactName": "artifactName1",

"artifactDescription": "artifactDescription1",

"artifactOwner": "artifactOwner1",

"artifactCreationDate": "1900-01-01T01:01:01Z",

"artifactModifiedDate": "1900-01-01T01:01:01Z",

"artifactArchivedDate": "1900-01-01T01:01:01Z",

"artifactDeletedDate": "1900-01-01T01:01:01Z",

"artifactCompressionType": "artifactCompressionType1",

"artifactArchived": "true",

"customData": {

"valuePair": null

}

},

"backImage": {

"artifactType": "artifactType1",

"externalId": "externalId1",

"artifact": "base64Binary Content",

"artifactName": "artifactName1",

"artifactDescription": "artifactDescription1",

"artifactOwner": "artifactOwner1",

"artifactCreationDate": "1900-01-01T01:01:01Z",

"artifactModifiedDate": "1900-01-01T01:01:01Z",

"artifactArchivedDate": "1900-01-01T01:01:01Z",

"artifactDeletedDate": "1900-01-01T01:01:01Z",

"artifactCompressionType": "artifactCompressionType1",

"artifactArchived": "true"

},

"accountId": "accountId1",

"status": "ExceedsDepositLimit",

"imageValidationIssueList": {

"imageValidationIssue": "CARMismatchFailed"

},

"remoteDepositNoteList": {

"currentPage": "1",

"totalPages": "1",

"pageSize": "1",

"positionIndex": "positionIndex1",

"returnPage": "1",

"returnPositionIndex": "returnPositionIndex1",

"note": {

"noteId": "noteId1",

"recordType": "Account",

"noteCode": "noteCode1",

"noteText": "noteText1",

"noteCreatedDateTime": "1900-01-01T01:01:01Z",

"noteExpirationDate": "1900-01-01",

"noteCreator": "noteCreator1"

}

},

"submittedDateTime": "1900-01-01T01:01:01Z",

"lastUpdateDateTime": "1900-01-01T01:01:01Z",

}

}

}

}

# Bibliography

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