

Artifact SPECIFICATION 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

# Authors and Change Log

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | | Changes |
| 0.0.1 |  | * Initial Creation | |
| 0.0.2 |  | * Cleanup of format and minor wording changes. | |
| 0.0.3 |  | * Convert document to use artifact.xsd, Error.xsd | |
| 0.0.4 |  | * Cleanup of typos | |
| 0.0.5 |  | * Updated Overview of Specification | |
| 0.0.6 |  | * Formatting of document * Remove information related to MAC parameters | |
| 0.0.7 |  | * Update artifactId to be able to have a combined key. Got rid of fiId in message. Added artifact at top level. | |
| 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 artifactMessage wrapper for every message to increase ability for infrastructure to serialize the data * Versioning and format change with release CUFX 3.0 | |
| 3.0 | **12/12/2013** | * 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, Added accountId, AccountIdlist, partyId, partyIdList to facilitate search and retrieval by these keys. * Added ArtifactUrI, transactionArtifactURI and TransactionArtificatId for referencing and or uri support for documents and images. | |

# Overview of Specification

The CUFX Artifact Data and Services specification defines the features of the CUFX Artifact service. The Artifact service provides a set of methods to read and write Artifacts such as Images and/or Documents to a Repository. This specification also contains sufficient data to describe how the object model may be added to other objects models (such as the party data model).

# Any know Errors in the document

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

Table of Contents

[Document Status 1](#_Toc532212254)

[Authors and Change Log 1](#_Toc532212255)

[Overview of Specification 2](#_Toc532212256)

[Any know Errors in the document 2](#_Toc532212257)

[Document Conventions 2](#_Toc532212258)

[Release 4.0 Global Update Notes 3](#_Toc532212259)

[Definitions related to the specification 3](#_Toc532212260)

[Concepts 4](#_Toc532212261)

[Data Elements 5](#_Toc532212262)

[Filters used when accessing the Artifact data 5](#_Toc532212263)

[Party Data attributes 5](#_Toc532212264)

[Services 5](#_Toc532212265)

[Artifact data 5](#_Toc532212266)

[Authenticaton 5](#_Toc532212267)

[Service Message: CREATE artifact Data 5](#_Toc532212268)

[REST-JSON Create artifact Example (Call and response) 6](#_Toc532212269)

[Service Message: Retrieve artifact Data by artifactId 6](#_Toc532212270)

[Service Message: Retrieve artifact by externalId 8](#_Toc532212271)

[REST-JSON Get Artifact by externalId Example (Call and response) 8](#_Toc532212272)

[Service Message: delete artifact by externalId 9](#_Toc532212273)

[REST-JSON Delete Artifact by artifactId Example (Call and response) 10](#_Toc532212274)

[Service Message: Update artifact by artifactId 11](#_Toc532212275)

[REST-JSON Update Artifact by artifactId Example (Call and response) 11](#_Toc532212276)

[General Error handling For All Services 12](#_Toc532212277)

[Bibliography 12](#_Toc532212278)

# 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

Artifact

A document or image that is stored in the Repository.

Document

A text, PDF file or other computer file format typically containing text and/or graphic images and/or formatting instructions.

Image

A graphic file containing digital data necessary for a computer system to render a digital picture.

Type

The document or image type such as PDF, JPG, PNG, GIF, DOC, etc. Typically this is the mime-type of the document.

Confidence Image

A special type of Image that is typically used to provide a user confidence in a system that they are about to provide sensitive information.

Signature Image

A special type of Image containing a digital representation of a person’s handwritten signature.

Repository

A system designed to store Artifacts and retrieve them upon demand.

Compression

An algorithm used to reduce the size of the Artifact by encoding information using fewer bits than the original representation.

# Concepts

Creation

This specification covers the concept of creating a new Artifact in the Repository.

DELETION

This specification covers the concept of deleting an Artifact in the Repository. It does not specify how the Repository should perform the delete (logical, physical etc.)

UPDATE

This specification covers the concept of updating an existing Artifact. It does not specify how the Repository should perform the update (i.e. by overwriting the existing entity or by versioning it)

ARCHIVE

This specification covers the concept of informing the Repository that a given Artifact may be archived. The Repository does not need to support this function.

Retrieval

This specification covers the concept of retrieving an Artifact from the Repository.

# Data Elements

## Filters used when accessing the Artifact 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.

## Party Data attributes

All CUFX fields related to a party are defined in **Artifact.xsd.**

Note: Fields not listed in the calling specification are not to be returned to the calling specification. i.e. If the field transaction type is not listed in the calling specification, then do not return the data field to alleviate issues with unexpected information and bloat of information being returned to light weight applications.

# Services

## Artifact data

To create, retrieve, update, and delete Artifact data in a Repository, clients access the artifact Data service.

|  |  |
| --- | --- |
| Definition | The artifact Data service defines the methods used to manipulate Artifacts in a Repository. |
| Overview of Capabilities | The artifact Data services supports creation, retrieval, and deletion of Artifacts in a Repository. |
| Dependencies | *Security Services* for authentication and security |
| CUFX REST LINK | https://api.dataprovider.com/artifact |
| CUFX WSDL LINK |  |

### Authenticaton

The typical use case is to provide user credentials with the artifactData request. After obtaining a valid Session Key, the client software will provide this Key with all subsequent requests. See Security Services for more detail. Requires artifactType, artifactCompressionType

### Service Message: CREATE artifact Data

|  |  |
| --- | --- |
| INPUTS | cufx:artifactMessage (which includes)   * [cufx:messageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * cufx:artifactFilter (read, update) * cufx:artifactList (for create, update, delete) |
| Outputs | cufx:artifactMessage (which includes)   * [cufx:messageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * cufx:artifactList |
| Return Values | cufx:artifactMessage (which includes)   * [cufx:messageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html)   + statusList |
| Service Specific Error Conditions | None: General Error Codes—see below |
| Side Effects | None |
| Dependencies | Security Services for authentication and security.  (REST-JSON) Handling null values in REST-JSON.  (Json.org) |

#### REST-JSON Create artifact Example (Call and response)

REQUEST:

Headers:

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

Accept: application/json

Accept-Charset: utf-8

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

X-API-Version: >=4.0.0

POST https://api.dataprovider.com/artifact

{

artifactMessage: {

“messageContext”: { <see messageContext.xsd>

},

“artifactList”: [{

“artifact”:{

“artifactType”: “xxxxxxxxxxx”, /\*---required---\*/

“externalId”: “xxxxxxxxxxx”, /\*---optional---\*/

“artifact”: “xxxxxxxxxxx”, /\*---required---\*/

“artifactName”: “xxxxxxxxxxx”, /\*---optional---\*/

  “artifactDescription”: “xxxxxxxxxxx”, /\*---optional---\*/

  “artifactOwner”: “xxxxxxxxxxx”, /\*---optional---\*/

“artifactCreationDate”: “xxxxxxxxxxx”, /\*---optional---\*/

“artifactModifiedDate”: “xxxxxxxxxxx”, /\*---optional---\*/

“artifactArchivedDate”: “xxxxxxxxxxx”, /\*---optional---\*/

“artifactDeletedDate”: “xxxxxxxxxxx”, /\*---optional---\*/

“artifactCompressionType”: “xxxxxxxxxxx”, /\*---required---\*/

“archived”: “xxxxxxxxxxx” /\*---optional---\*/

}]

}

 }

The Repository responds to artifact requests with a JSON-formatted message:

RESPONSE:

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

artifactMessage: {

“messageContext”: { <see messageContext.xsd>

},

“artifactList”: {

“artifact”:{

  “artifactId”:{“artifactUniqueId”:”1234…”}

}

}

}

### Service Message: Retrieve artifact Data by artifactId

|  |  |
| --- | --- |
| INPUTS | cufx:artifactMessage (which includes)   * [cufx:messageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * cufx:artifactFilter |
| Outputs | cufx:artifactMessage (which includes)   * [cufx:messageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * cufx:artifactList |
| Return Values | cufx:artifactMessage (which includes)   * [cufx:messageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html)   + statusList |
| Service Specific Error Conditions | None: General Error Codes—see below |
| Side Effects | None |
| Dependencies | Security Services for authentication and security.  (REST-JSON) Handling null values in REST-JSON.  (Json.org) |
| CUFX REST LINK | https://api.dataprovider.com/artifact |
| CUFX WSDL LINK |  |
| Testing Procedures for Certification | Access the CUFX artifact Data test service and verify that test data is returned appropriately. |
| Derivative Services | None |

Clients submit an artifact data request to the repository’s artifact URL. For all CUFX RESTful-JSON requests, the client must set these headers:

**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.dataprovider.com/artifact>

{

artifactMessage:{

“messageContext”: { <see messageContext.xsd>

},

“artifactFilter”: {

“artifactId”:{“artifactUniqueId”:”1234…”}

}

}

}

The Repository responds to artifact requests with a JSON-formatted message:

**RESPONSE:**

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

artifactMessage:{

“messageContext”: { <see messageContext.xsd>

},

"artifactList": {

  “artifact”: {

  “artifactId”:{“artifactUniqueId”:”1234…”}

“artifactType”: MIME type of the Artifact

“externalId”: reference value used by the consumer

“artifact”: binary object containing the digital data

“artifactName”: name of the Artifact

  “artifactDescription”: description of the Artifact

  “artifactOwner”: creator of the Artifact

“artifactCreationDate”: creation date of the Artifact

“artifactModifiedDate”: modified date of the Artifact

“artifactArchivedDate”: archived date of the Artifact

“artifactDeletedDate”: deleted date of the Artifact

“artifactCompressionType”: compression type of the Artifact

“archived”: archived status of the Artifact

}

}

}

### Service Message: Retrieve artifact by externalId

|  |  |
| --- | --- |
| INPUTS | cufx:artifactMessage (which includes)   * [cufx:messageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * cufx:artifactFilter |
| Outputs | cufx:artifactMessage (which includes)   * [cufx:messageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * cufx:artifactList |
| Return Values | cufx:artifactMessage (which includes)   * [cufx:messageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html)   + statusList |
| Service Specific Error Conditions | None: General Error Codes—see below |
| Side Effects | None |
| Dependencies | Security Services for authentication and security.  (REST-JSON) Handling null values in REST-JSON.  (Json.org) |
| CUFX REST LINK | https://api.dataprovider.com/artifact |
| CUFX WSDL LINK |  |
| Testing Procedures for Certification | Access the CUFX artifact Data test service and verify that test data is returned appropriately. |
| Derivative Services | None |

#### REST-JSON Get Artifact by externalId Example (Call and response)

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.dataprovider.com/artifact>

{

artifactMessage: {

“messageContext”: { <see messageContext.xsd>

},

“artifactFilter”: {

“artifactList”: {

“externalId”: “xxxxxxxxxxx” /\*---required---\*/

}

}

}

 }

The Repository responds to artifact requests with a JSON-formatted message:

RESPONSE:

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

artifactMessage:{

“messageContext”: { <see messageContext.xsd>

},

“artifactList”: {

  “artifact”:

  {

  “artifactId”:{“artifactUniqueId”:”1234…”}

“artifactType”: MIME type of the Artifact

“externalId”: reference value used by the consumer

“artifact”: binary object containing the digital data

“artifactName”: name of the Artifact

  “artifactDescription”: description of the Artifact

  “artifactOwner”: creator of the Artifact

“artifactCreationDate”: creation date of the Artifact

“artifactModifiedDate”: modified date of the Artifact

“artifactArchivedDate”: archived date of the Artifact

“artifactDeletedDate”: deleted date of the Artifact

“artifactCompressionType”: compression type of the Artifact

“archived”: archived status of the Artifact

}

}

}

### Service Message: delete artifact by externalId

|  |  |
| --- | --- |
| INPUTS | cufx:artifactMessage (which includes)   * [cufx:messageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * cufx:artifactlist |
| Outputs | cufx:artifactMessage (which includes)   * [cufx:messageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) |
| Return Values | cufx:artifactMessage (which includes)   * [cufx:messageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html)   + statusList |
| Service Specific Error Conditions | None: General Error Codes—see below |
| Side Effects | None |
| Dependencies | Security Services for authentication and security.  (REST-JSON) Handling null values in REST-JSON.  (Json.org) |
| CUFX REST LINK | https://api.dataprovider.com/artifact |
| CUFX WSDL LINK |  |
| Testing Procedures for Certification | Access the CUFX artifactData test service and verify that test data is deleted appropriately. |
| Derivative Services | None |

#### REST-JSON Delete Artifact by artifactId Example (Call and response)

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.dataprovider.com/artifact>

{

artifactMessage:{

“messageContext”: { <see messageContext.xsd>

},

 “artifactList”: {

Artifact: {

“artifactId”:{“artifactUniqueId”:”1234…”}

}

}

}

 }

The Repository responds to artifact requests with a JSON-formatted message:

RESPONSE:

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

“artifactMessage”: {

"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": {

"status": {

"statusType": "Success"

}

},

},

      }

}

### Service Message: Update artifact by artifactId

|  |  |
| --- | --- |
| INPUTS | cufx:artifactMessage (which includes)   * [cufx:messageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * cufx:artifactFilter * cufx:artifactlist |
| Outputs | cufx:artifactMessage (which includes)   * [cufx:messageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * cufx:artifactlist |
| Return Values | cufx:artifactMessage (which includes)   * [cufx:messageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html)   + statusList |
| Service Specific Error Conditions | None: General Error Codes—see below |
| Side Effects | None |
| Dependencies | Security Services for authentication and security.  (REST-JSON) Handling null values in REST-JSON.  (Json.org) |
| CUFX REST LINK | https://api.dataprovider.com/artifact |
| CUFX WSDL LINK |  |
| Testing Procedures for Certification | Access the CUFX artifact Data test service and verify that test data is updated appropriately. |
| Derivative Services | None |

#### REST-JSON Update Artifact by artifactId Example (Call and response)

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.dataprovider.com/artifact>

{

artifactMessage:{

“messageContex”: { <see messageContext.xsd>

},

“artifactList: {[

“artifact”:{

  “artifactId”:{“artifactUniqueId”:”1234…”}

“artifactType”: “xxxxxxxxxxx”, /\*---optional---\*/

“externalId”: “xxxxxxxxxxx”, /\*---optional---\*/

“artifact”: “xxxxxxxxxxx”, /\*---optional---\*/

“artifactName”: “xxxxxxxxxxx”, /\*---optional---\*/

  “artifactDescription”: “xxxxxxxxxxx”, /\*---optional---\*/

  “artifactOwner”: “xxxxxxxxxxx”, /\*---optional---\*/

“artifactCreationDate”: “xxxxxxxxxxx”, /\*---optional---\*/

“artifactModifiedDate”: “xxxxxxxxxxx”, /\*---optional---\*/

“artifactArchivedDate”: “xxxxxxxxxxx”, /\*---optional---\*/

“artifactDeletedDate”: “xxxxxxxxxxx”, /\*---optional---\*/

“artifactCompressionType”: “xxxxxxxxxxx”, /\*---optional---\*/

“archived”: “xxxxxxxxxxx” /\*---optional---\*/

}

]}

}

 }

The Repository responds to artifact requests with a JSON-formatted message:

RESPONSE:

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

“artifactMessage”: {

"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": {

"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.