

Contact 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

# Authors and Change Log

|  |  |  |
| --- | --- | --- |
| Version | Date | Changes |
| 0.0.01 |  | * Initial Draft |
| 0.0.02 |  | * Updated Overview of Specification |
| 0.0.03 |  | * Minor updates and correction of typos |
| 0.0.04 |  | * Update XSD filenames to PascalCase |
| 0.0.05 |  | * Update examples to be consistent with current XSD |
| 0.0.06 |  | * Added “point” in structure and fine-tuned examples with current XSD. |
| 0.0.07 |  | * Switch to use X-HTTP-METHOD-OVERRIDE standard rather than subMethod non-Standard method for overriding request types. * Create a contactMessage wrapper for every message to increase ability for infrastructure to serialize the data |
| 3.0 | **10/29/2013** | * 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 | **12/20/2013** | * Review and updated typos, formatting. |
| 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, added 'Other' as address type as catch all, Split out AddressType to SimpleType for consistency from Contact Address, Split out simpletypes EmailType and PhoneType for consistency. Replaced point xs:group with complexType. Updated element 'types' to be more specific matching complex name. |
| 4.1 | **12/10/2018** | * Updated to release 4.1, Corrected spelling error in desc, added contactPointDescription, corrected / improved documentation. |

# Overview of Specification

The CUFX contact data model and services defines the contact data model and services for party and relationship. Contacts include contact points such as email, mail, phone, instant message, and social media.

# Any knowN Errors in the document

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

# Table of Contents

[Document Status 1](#_Toc532212992)

[Authors and Change Log 1](#_Toc532212993)

[Overview of Specification 2](#_Toc532212994)

[Any knowN Errors in the document 2](#_Toc532212995)

[Table of Contents 2](#_Toc532212996)

[Document Conventions 2](#_Toc532212997)

[Release 4.0 Global Update Notes 3](#_Toc532212998)

[Definitions related to the specification 3](#_Toc532212999)

[Data Elements 3](#_Toc532213000)

[Filters used when accessing the Contact data 3](#_Toc532213001)

[Contact Data attributes 4](#_Toc532213002)

[Party Services 4](#_Toc532213003)

[Overview 4](#_Toc532213004)

[Contact Resource based create, read, update, delete services 4](#_Toc532213005)

[REST-JSON CREATE Contact example 5](#_Toc532213006)

[REST-JSON READ Contact example 6](#_Toc532213007)

[REST-JSON READ Contact example 7](#_Toc532213008)

[REST-JSON UPDATE Contact example 8](#_Toc532213009)

[REST-JSON DELETE Contact example 9](#_Toc532213010)

[General Error handling For All Services 10](#_Toc532213011)

[Bibliography 10](#_Toc532213012)

# 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 utilizes Word Styles.

All Citations must utilize Word Citations so that it automatically shows at the end of the document.

All updates after the initial creation must be performed using Tracking Changes turned 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

**Contact**

Any contact point for a person or entity which may have an account relationship with the financial institution and stored in a financial services platform. Most contacts will be created, read, updated, or deleted using the party services. This is just provided as an additional method for updating contact information if necessary.

# Data Elements

## Filters used when accessing the Contact data

Refer to Security Services documentation to understand what may be contained in 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 **ContactFilter.xsd.** This is useful for pulling back or updating specific contacts, contacts associated to a party, relationship, etc. and, when combined with contact types, for example, all email addresses for a party, relationship, etc.

## Contact Data attributes

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

Note: Fields not listed in the calling specification are not to be returned to the calling specification. That is, 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 or a bloat of information being returned to lightweight applications.

# Party Services

## Overview

|  |  |
| --- | --- |
| Definition | Collection of services to manage a contact |
| Overview of Capabilities | Create, read, update and delete a contact. The following scenarios may exist. The contact may be connected to an existing lead or party. In addition, a contact may exist without an existing party or lead in preparation to be connected to a party or lead. Some systems may need to place the contact in a temporary location until the party or lead is made available. |
| Dependencies | Security Services, messageContext, party |
| Sample CUFX REST LINK | https://api.dataprovider.com/contact/ |
| CUFX SOAP LINK |  |
| CUFX WaDL LINK |  |

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

|  |  |
| --- | --- |
| INPUTS | cufx:ContactMessage (which includes…)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * cufx:contactFilter (for read, update) * cufx:contactList (for create, update, delete) |
| Outputs | cufx:ContactMessage (which includes…)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * cufx:contactList |
| Return Values | cufx:ContactMessage (which includes…)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html)   + statusList |
| Side Effects | Creation, update or deletion of contact; 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 ContactFilter.xsd  Attributes: contactList : See Contact.xsd |

### REST-JSON CREATE Contact example

This example shows where the party is not known.

Note: Not all fields are listed for simplicity of an example to create a contact. This example creates an email contact.

**Required**: messageContext, at least one contact within contactList

**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/contact

{

“contactMessage”:{

“messageContext”: { <see MessageContext.xsd>

},

“contactList”: [

{“contactType”:”Email”,

“point”:{

“email”:{

“type”:”Home”,

“address”:[tom@barnum.com](mailto:tom@barnum.com)

}

},

“timeOfDay”:”Afternoon”,

“demonstratedAccess”:{

“dateTime”:”2013-04-30T12:05Z”,

“fullName”:“Tom Ruben Thumb”,

“username”:”tthumb”

}

}

]

}

}

**RESPONSE:**

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{“contactMessage”:{

“messageContext”: { <see MessageContext.xsd>

},

“contactList”: [

{“contactId”:”15345515115df”,

“contactType”:”Email”,

“point”:{

“email”:{

“type”:”Home”,

“address”:[tom@barnum.com](mailto:tom@barnum.com)

}

},

“timeOfDay”:”Afternoon”,

“demonstratedAccess”:{

“dateTime”:”2013-04-30T12:05Z”,

“fullName”:“Tom Ruben Thumb”,

“username”:”tthumb”

}

}

]

}

}

### REST-JSON READ Contact example

This example reads a specific contact record.

Note: Not all fields are listed for simplicity of an example to read a contact.

**Required**: messageContext, at least one valid filter in contactFilter

**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/contact

{

“contactMessage”:{

“messageContext”: { <see MessageContext.xsd> },

“contactFilter”:{

“contactIdList”:[

”15345515115df”

]

}

}

}

**RESPONSE:**

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{“contactMessage”:{

“messageContext”: { <see MessageContext.xsd>

},

“contactList”: [

{“contactId”:”15345515115df”,

“contactType”:”Email”,

“point”:{

“email”:{

“type”:”Home”,

“address”:[tom@barnum.com](mailto:tom@barnum.com)

}

},

“timeOfDay”:”Afternoon”,

“demonstratedAccess”:{

“dateTime”:”2013-04-30T12:05Z”,

“fullName”:“Tom Ruben Thumb”,

“username”:”tthumb”

}

}

]

}

}

### REST-JSON READ Contact example

Note: Not all fields are listed for simplicity of an example to read a contact. This example shows how to read all contacts for a specific party.

**Required**: messageContext, at least one valid filter in contactFilter

**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/contact

{

“contactMessage”:{

“messageContext”: { <see MessageContext.xsd> },

“contactFilter”:{

“partyIdList”:[

”151349885h348870ag”

]

}

}

}

**RESPONSE:**

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{“contactMessage”:{

“messageContext”: { <see MessageContext.xsd>

},

“contactList”: [

{“contactId”:”15345515115df”,

“contactType”:”Email”,

“point”:{

“email”:{

“type”:”Home”,

“address”:[tom@barnum.com](mailto:tom@barnum.com)

}

},

“timeOfDay”:”Afternoon”,

“demonstratedAccess”:{

“dateTime”:”2013-04-30T12:05Z”,

“fullName”:“Tom Ruben Thumb”,

“username”:”tthumb”

}

},

{“contactId”:”84hrt3654967845”,

“contactType”:”Address”,

“point”:{

“address”:{

“type”:”Home”,

“line1”:”123 Main St”,

“line2”:”Apt 1A”,

“city”:”Anywhere”,

“stateProvince”:”FL”,

“postalCode”:”35999”,

“countryCode”:”USA”

}

}

}

}

}

### REST-JSON UPDATE Contact example

Note: Not all fields are listed for simplicity of an example to update a contact. This example updates an email contact.

**Required**: messageContext, at least one valid filter in partyFilter, the specific id of the party to be updated in the message

**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/contact

{

“contactMessage”:{

“messageContext”: { <see MessageContext.xsd>

},

“contactFilter”:{

“contactIdList”:[

”15345515115df”

]

}

“contactList”: [

{“contactId”:”15345515115df”,

“point”:{

“email”:{

“address”:”[iquit@bigandtall.com](mailto:iquit@bigandtall.com)”

}

}

}

]

}

}

**RESPONSE:**

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{“contactMessage”:{

“messageContext”: { <see MessageContext.xsd>

},

“contactList”: [

{“contactId”:”15345515115df”

“contactType”:”Email”,

“point”:{

“email”:{

“type”:”Home”,

“address”:”iquit@bigandtall.com”

}

}

]

}

}

### REST-JSON DELETE Contact example

Note: Not all fields are listed for simplicity of an example to delete a contact. This example deletes all contacts for a party.

**Required**: messageContext, at least one valid filter in partyFilter, the specific id of the party to be updated in the message

**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/party

{

“contactMessage”:{

“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"

},

},

“contactFilter”:{

“partyIdList”:[

”151349885h348870ag”

]

}

}

}

**RESPONSE:**

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

"contactMessage": {

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

"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

*E.164.* (n.d.). Retrieved 06 28, 2012, from International PUblic Telecommunications Number Plan: http://www.itu.int/rec/T-REC-E.164/en

*North American Number Plan Administration.* (n.d.). Retrieved 06 28, 2012, from North American Number Plan Administration: http://www.nanpa.com/

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