

preference Data and Services 4.5

# 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 |
| 0.0.01 |  | * Initial Creation |
| 0.0.02 |  | * Added filter variables * Cleaned up formatting. * Restructured order and one-to-many relationships within preference data to make it more flexible and clear. * Moved demonstrated access to party contact data * Shortened many of the value names because they were getting too long. |
| 0.0.03 |  | * Reorganized Key values to other tables to use direct key names rather than logical key name |
| 0.0.04 |  | * Replaced spaces in constants in domains with underscores * Added example for pulling by list of account ID’s. * Renumbered document based on correct number scheme |
| 0.0.05 |  | * Made more consistent with other specs in formatting of constants with underscores * Removed definitions for field level data as it’s been moved to preference.xsd * Replace parameters with messageContext.xsd and preferenceFilter.xsd definitions * Updated error section to refer to Error documentation. |
| 0.0.06 |  | * Fix typos in services |
| 0.0.07 |  | * Update plurals to use List |
| 0.0.08 |  | * Fix small typos in definition |
| 0.0.09 |  | * Updated Overview of Specification |
| 0.0.10 |  | * Update XSD filenames to PascalCase |
| 0.0.11 |  | * Update id lists to be consistent with XSD |
| 0.0.12 |  | * Make type and subtype be a tiered structure based on parent type. * Add template preference example. |
| 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 preferenceMessage 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, Added PaperStatement enum, comment change for EStatement enum, Added M2Mtransfers and P2Ptransfers enabled preference types, |
| 4.1 | **12/10/2018** | * Updated to release 4.1, Global removal of choice constructs to eliminate cross language serialization issues. |
| 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. |
| 4.3 | **10/07/2019** | * Updated to release 4.3, renamed file removing version as proper version control is being used in Github. |
| 4.4 | **10/20/2020** | * Updated to release 4.4, Preference - Added valueDataType, accountIdentificationList to Preference * Added enum NoSolicitation to PreferenceType * Added PreferenceSettingList and complex type PreferenceSetting * Added type DataType * PreferenceFilter - Added accountIdentificationList to filter |
| 4.5 | **04/02/2021** | * Updated to release 4.5 |

# Overview of Specification

The CUFX preference Data and Services is a collection of services that drive the user experience and allow services to modify themselves and to be personalized to the consumer of the service. This model is flexible in that it can point to multiple accounts, contact points, parties, relationship, etc.

# Any know Errors in the document

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

# Table of Contents

[Document Status 1](#_Toc68101336)

[Change Log 1](#_Toc68101337)

[Overview of Specification 2](#_Toc68101338)

[Any know Errors in the document 2](#_Toc68101339)

[Table of Contents 2](#_Toc68101340)

[Document Conventions 3](#_Toc68101341)

[CUFX API and Documentation Support 3](#_Toc68101342)

[Release 4.0 Global Update Notes 3](#_Toc68101343)

[Release 4.4 Global Update Notes 4](#_Toc68101344)

[Definitions related to the specification 4](#_Toc68101345)

[Data Elements 4](#_Toc68101346)

[Filters used when accessing the Preference data 5](#_Toc68101347)

[Preference Data attributes 5](#_Toc68101348)

[Services 5](#_Toc68101349)

[Overview 5](#_Toc68101350)

[Preference Resource based create, read, update, delete services 5](#_Toc68101351)

[REST-JSON READ PREFERENCE template list of available EAlerts. 6](#_Toc68101352)

[REST-JSON CREATE PREFERENCE 7](#_Toc68101353)

[REST-JSON READ PREFERENCE of a specific preference ID example 8](#_Toc68101354)

[REST-JSON READ PREFERENCE preferences for a list of accounts 9](#_Toc68101355)

[REST-JSON UPDATE PREFERENCE for with specific Preference ID. 10](#_Toc68101356)

[REST-JSON DELETE PREFERENCE example. 11](#_Toc68101357)

[General Error handling For All Services 12](#_Toc68101358)

[Bibliography 12](#_Toc68101359)

# 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.

# Definitions related to the specification

preference

Preference is a collection of attributes that drive the user experience and allow services to modify themselves and to be personalized to the consumer of the service. This model is flexible in that it can point to multiple accounts, contact points, parties, relationship, etc.

# Data Elements

## Filters used when accessing the Preference 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 P**referenceFilter.xsd.**

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

## Preference Data attributes

All CUFX fields related to a preference are defined in **Preference.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

## Overview

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

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

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

### REST-JSON READ PREFERENCE template list of available EAlerts.

This example pulls all available template records so system knows what available alerts the party can set up.

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

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

**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: >=4.5.0

POST https://api.datasource.com/preferencemessage

{

“preferenceMessage”:{

“messageContext”: { <see MessageContext.xsd>

},

“**preferenceFilter**”:{

“**preferenceStatusList**”:[ ”Template” ],

“preferenceTypeList”:[“EAlert” ]

}

}

}

**RESPONSE**:

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

“preferenceMessage”:{

“messageContext”: { <see MessageContext.xsd>

},

“preferenceList”:[{

“preference”: [

{“**preferenceId**”:”12334”,

“preferenceType”:”EAlert”,

“subType”:{“eAlertSubType” :”AtmWithdrawalExceeds”},

“preferenceStatus”:”Template”,

“value”:”500.00”

}],

“preference”: [

{“**preferenceId**”:”12335”,

“preferenceType”:”EAlert”,

“subType”:{“eAlertSubType”:”AtmDepositExceeds”},

“value”:”100.00”,

“preferenceStatus”:”Template”

}],

“preference”: [

{“**preferenceId**”:”12336”,

“preferenceType”:”EAlert”,

“subType”:{“eAlertSubType”:”PasswordResetFailure”},

“preferenceStatus”:”Template”

}],

…

]

}

}

### REST-JSON CREATE PREFERENCE

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 preference.

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

**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: >=4.5.0

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

{

“preferenceMessage”:{

“messageContext”: { <see MessageContext.xsd>

},

“preferenceList”:[

“preference”: [

{“partyIdList”: [ ”15353534155asdf5” ]

“relationshipIdList: [ ”168695596adf” ],

“preferenceType”:”EAlert”,

“subType”:{“AlertSubType”:”AtmWithdrawalExceeds”},

“preferenceStatus”:”Active”,

“value”:”500.00”

}],

“preference”: [

{“partyIdList”: [ ”1asdf6869984651631f” ],

“preferenceType”:”WebsiteFormatStyleSheet”,

“preferenceStatus”:”Active”,

“value”:”splash.css”

}],

“preference”: [

{“contactIdList”: [ ”15353534155asdf5” ],

“preferenceType”:” EAlert”,

“subType”:{“eAlertSubType” :”ScheduledMessage”},

“value”:”2013-05-15T12:30Z”,

“preferenceStatus”:”Active”,

“alertCustomText”:”Remember to pay your car bill!”

}]

]

}

}

RESPONSE:

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

“preferenceMessage”:{

“messageContext”: { <see MessageContext.xsd>

},

“preferenceList”:[

“preference”: [

{“preferenceId”:”7070615644aasdfasdfa535478932sf”,

{“partyIdList”: [ ”15353534155asdf5” ]

“relationshipIdList: [ ”168695596adf” ],

“preferenceType”:”EAlert”,

“subType”:{“eAlertSubType” :”AtmWithdrawalExceeds”},

“preferenceStatus”:”Active”,

“value”:”500.00”

}],

“preference”: [

{“preferenceId”:”707061515vg4143asdfa5354125341b”,

“partyIdList”: [ ”1asdf6869984651631f” ],

“subType”:{“eAlertSubType” :”AtmWithdrawalExceeds”},

“preferenceStatus”:”Active”,

“value”:”splash.css”

}],

“preference”: [

{“preferenceId”:”1525731515vg425756fa53541224abd”,

“contactIdList”: [”15353534155asdf5” ],

“preferenceType”:”EAlert”,

“subType”:{“eAlertSubType” :”ScheduledMessage”},

“preferenceStatus”:”Active”,

“value”:”2013-05-15T12:30Z”,

“alertCustomText”:”Remember to pay your car bill!”

}]

]

}

}

### REST-JSON READ PREFERENCE of a specific preference ID example

This example shows where the preference ID is known.

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

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

**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: >=4.5.0

POST https://api.datasource.com/preferencemessage

{

“preferenceMessage”:{

“messageContext”: { <see MessageContext.xsd>

},

“**preferenceFilter**”:{

“**preferenceIdList**”:[ ”7070615644aasdfasdfa535478932sf” ]

}

}

}

**RESPONSE**:

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

“preferenceMessage”:{

“messageContext”: { <see MessageContext.xsd>

},

“preferenceList”:[

“preference”: [

{“**preferenceId**”:”7070615644aasdfasdfa535478932sf”,

“**partyIdList**”: [ ”15353534155asdf5” ]

“relationshipIdList: [ ”168695596adf” ],

“preferenceType”:”EAlert”,

“subType”:{“eAlertSubType” :”AtmWithdrawalExceeds”},

“preferenceStatus”:”Active”,

“value”:”500.00”

}]

]

}

}

### REST-JSON READ PREFERENCE preferences for a list of accounts

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

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

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

**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: >=4.5.0

POST https://api.datasource.com/preferencemessage

{

“preferenceMessage”:{

“messageContext”: { <see MessageContext.xsd>

},

“**preferenceFilter**”:{

“accountIdList”:[

“615644aa32sf”,

”15345151s25”,

”1513341635”

]

]

}

}

**RESPONSE**:

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

“preferenceMessage”:{

“messageContext”: { <see MessageContext.xsd>

},

“preferenceList”:[

“preference”: [

{“**preferenceId**”:” 7070615644aasdfasdfa535478932sf”,

“accountIdList”:[ ”615644aa32sf” ],

“preferenceType”:”EAlert”,

“subType”:{“eAlertSubType” :”AtmWithdrawalExceeds”},

“preferenceStatus”:”Active”,

“value”:”500.00”

}],

“preference”: [

{“**preferenceId**”:”8415151615644adsfdfasdfa535sdf93”

“accountIdList”:[ “15345151s25”],

“preferenceType”:”EAlert”,

“subType”:{“eAlertSubType” :”AtmWithdrawalExceeds”},

“preferenceStatus”:”Active”,

“value”:”250.00”

}],

“preference”: [

{“**preferenceId**”:”9151061561507134had788451”

“accountIdList”:[ ”15345151s25” ],

“preferenceType”:”EAlert”,

“subType”:{“eAlertSubType” :”AtmWithdrawalExceeds”},

“preferenceStatus”:”Active”,

“value”:”1000.00”

}],

]

}

}

### REST-JSON UPDATE PREFERENCE for with specific Preference ID.

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

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

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

**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: >=4.5.0

PUT https://api.datasource.com/preferencemessage

{

“preferenceMessage”:{

“messageContext”: { <see MessageContext.xsd>

},

“**preferenceFilter**”:{

“**preferenceIdList**”: [ ”7070615644aasdfasdfa535478932sf” ]

}

“preferenceList”:[

“preference”: [

{“**preferenceId**”:”7070615644aasdfasdfa535478932sf”,

“value”:”2000.00”

}

]

]

}

}

**RESPONSE**:

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

“preferenceMessage”:{

“messageContext”: { <see MessageContext.xsd>

},

“preferenceList”:[

“preference”: [

{“**preferenceId**”:”7070615644aasdfasdfa535478932sf”,

“**partyIdList**”: [ ”15353534155asdf5” ],

“relationshipIdList: [ ”168695596adf” ],

“preferenceType”:”EAlert”,

“subType”:{“eAlertSubType” :”AtmWithdrawalExceeds”},

“preferenceStatus”:”Active”,

“value”:”2000.00”

}]

]

}

}

### REST-JSON DELETE PREFERENCE example.

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

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

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

**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: >=4.5.0

PUT https://api.datasource.com/preferencemessage

{

“preferenceMessage”:{

“messageContext”: { <see MessageContext.xsd>

},

“**preferenceFilter**”:{

“accountIdList”:[

615644aa32sf”,

”15345151s25”,

”1513341635”

]

]

}

}

**RESPONSE**:

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

“preferenceMessage”:{

“messageContext”: { <see MessageContext.xsd>

}

}

}

# 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.