

card Data 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)

# Authors and Change Log

|  |  |  |
| --- | --- | --- |
| Version | Date | Changes |
| 0.0.01 |  | * Initial Draft |
| 0.0.02 |  | * Added page number in footer and other minor typos |
| 0.0.03 |  | * Updated Overview of Specifications |
| 0.0.04 |  | * Minor correction on verbiage and typos |
| 0.0.05 |  | * Make examples consistent with current XSD |
| 0.0.06 |  | * Added more fields to support additional use cases |
| 0.0.07 |  | * Switch to use X-HTTP-METHOD-OVERRIDE standard rather than subMethod non-Standard method for overriding request types. * Create a cardMessage wrapper for every message to increase ability for infrastructure to serialize the data * Changed return item for Create to cardList rather than card to enable multiple cards to be created in one call. |
| 3.0 | **10/29/2013** | * Versioning and format change with release CUFX 3.0 |
| 3.0 | **12/12/2013** | * Update examples X-API-Version to >=3.0.0 |
| 3.0 | **04/02/2014** | * Updated card data attributes element cardDesignImageArtifactId |
| 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 cardAddress to support passing fields directly when reference is not available. Add accountStatusList to filter. Added activation, expiration, and blocked DateRange filters. |
| 4.1 | **12/10/2018** | * Updated to release 4.1, * Card - Updated CardStatus enums, added suspended status and suspendedDateTime. Removed suspendDateTime, added cardStatusDateTime, cardStatusReason. Deprecated blockedDateTime and blockedreason. Added cardstatusList * Card, CardFilter - Added cardstatusList to card, cardfilter. |
| 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. * Card -Added customData to Card object. * Card. Card Filter - Added cardTypeList to cardFilter |
| 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, Card - Added accountType and accountSubType to Card. * CardFilter - Added accountIdentificationList. |

# Overview of Specification

The Card Data and Services specification defines the card data object for use by all specifications. A card defines a card type of ATM, credit, debit or prepaid. A card connects information to deposit accounts, loan accounts and associated parties. This service is used to create, read, update and delete a card.

# Any known Errors in the document

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

# Table of Contents

[Document Status 1](#_Toc54099886)

[Authors and Change Log 1](#_Toc54099887)

[Overview of Specification 2](#_Toc54099888)

[Any known Errors in the document 2](#_Toc54099889)

[Table of Contents 2](#_Toc54099890)

[Document Conventions 3](#_Toc54099891)

[CUFX API and Documentation Support 3](#_Toc54099892)

[Release 4.0 Global Update Notes 3](#_Toc54099893)

[Release 4.4 Global Update Notes 4](#_Toc54099894)

[Definitions related to the specification 4](#_Toc54099895)

[Data Elements 4](#_Toc54099896)

[Filters used when accessing the card data 4](#_Toc54099897)

[card Data attributes 4](#_Toc54099898)

[CARD Services 5](#_Toc54099899)

[Overview 5](#_Toc54099900)

[Card Resource based create, read, update, delete services 5](#_Toc54099901)

[REST-JSON CREATE Card example 6](#_Toc54099902)

[REST-JSON READ Card example 7](#_Toc54099903)

[REST-JSON UPDATE Card example 8](#_Toc54099904)

[REST-JSON DELETE Card example 9](#_Toc54099905)

[REST-JSON ACTIVATE Card example 11](#_Toc54099906)

[REST-JSON BLOCK Card example 12](#_Toc54099907)

[General Error handling For All Services 14](#_Toc54099908)

[Bibliography 14](#_Toc54099909)

# 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

**card**

A plastic card of any type that allows access to one or more deposit or loan accounts by a specified party.

# Data Elements

## Filters used when accessing the card 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. Include any filter variables related to the request. See **CardFilter.xsd.**

## card Data attributes

All CUFX fields related to a card are defined in card.xsd. A summary of the attributes is listed here for reference.

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.

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| cardId | A unique identifier for the card. |
| cardNumber | The number of the card. |
| cardType | The type of the card. One of: ATM, Credit, Debit, Prepaid |
| cardSubType | The flavor of the card (e.g. platinum, sports team branded, etc.) |
| expirationDate | The expiration date for the card |
| pin | The card’s pin number |
| cvv2 | The verification code for the card |
| linkedAccountList | A list of accounts that are linked to the card |
| partyId | The party whose name appears on the card |
| overrideAddressContactId | Overrides the address supplied by the party |
| nameOnCard | The name(s) on the card. |
| activationDateTime | The date and time on which the card was activated |
| cardStatus | The status of the card. On of: Inactive, Active, Blocked |
| blockedReason | If the card is blocked, specifies the reason for the block (e.g. lost, stolen) |
| blockedDateTime | If the card is blocked, specifies the date and time on which the card was blocked |
| virtualNumber | A virtual number associated with the card |
| merchantCountryCodes | A list of country codes at which the card is allowed to be used |
| cardDesignImageArtifactId | Specifies the image (via an ArtifactId) for the card’s design |

# CARD Services

## Overview

|  |  |
| --- | --- |
| Definition | Collection of services to manage a card |
| Overview of Capabilities | Create, read, update and delete a card. The following scenarios may exist. The card may be connected to an existing party and one or more account(s). |
| Dependencies | Security Services, messageContext |
| Sample CUFX REST LINK | https://api.dataprovider.com/cardmessage |
| CUFX SOAP LINK |  |
| CUFX WaDL LINK |  |

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

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

### REST-JSON CREATE Card example

Note: Not all fields are listed for simplicity of an example to create a card. This example creates a Debit Card.

**Required**: messageContext, card.

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.datasource.com/cardmessage

{

“cardMessage”:{

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

“cardList”:[

{

“cardType”: ”Debit”,

“cardSubType”: “NFL”,

“linkedAccountList”:

[

{

“id”:”283746927364”,

“priority”:”1”

},

{

“id”:”7234234565”,

“priority”:”2”

}

],

“partyId”:”23654283”

}

]

}

}

**RESPONSE:**

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

“cardMessage”:{

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

“cardList”:

{

“cardId”: “87923423467qq”,

“cardType”: ”Debit”,

“cardSubType”: “NFL”,

“linkedAccountList”:

[

{

“id”:”283746927364”,

“priority”:”1”

},

{

“id”:”7234234565”,

“priority”:”2”

}

],

“partyId”:”23654283”

}

}

}

}

### REST-JSON READ Card example

Note: Not all fields are listed for simplicity of an example to read a card. This example reads a Debit Card.

**Required**: messageContext, at least one valid filter in cardFilter.

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: >=4.3.0

**POST** [**h**ttps://api.datasource.com/card](https://api.datasource.com/card)message

{

“cardMessage”:{

“messageContext”: { <see messageContext.xsd>

},

“cardFilter”:{

“cardIdList”:[

”87923423467qq”

]

}

}

}

**RESPONSE:**

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

“cardMessage”:{

“messageContext”: { <see messageContext.xsd>

},

“cardList”: {

{

“cardId”: “87923423467qq”,

“cardType”: ”Debit”,

“cardSubType”: “NFL”,

“linkedAccountList”:

[

{

“accountId”:”283746927364”,

“priority”:”1”

},

{

“accountId”:”7234234565”,

“priority”:”2”

}

],

“partyId”:”23654283”

}

}

}

}

### REST-JSON UPDATE Card example

Note: Not all fields are listed for simplicity of an example to update a card. This example updates a Debit Card.

**Required**: messageContext, at least one valid filter in cardFilter, the specific id of the card 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: >=4.3.0

**PUT h**ttps://api.datasource.com/cardmessage

{

“cardMessage”: {

“messageContext”: { <see messageContext.xsd>

},

“cardFilter”:{

“cardIdList”:[

”87923423467qq”

]

},

“cardList”: [

{

“cardId”: “87923423467qq”,

“cardType”:”Debit”,

“cardSubType”: “NFL”,

“linkedAccountList”:

[

{

“accountId”:”283746927364”,

“priority”:”1”

},

{

“accountId”:”5qw387987”,

“priority”:”2”

},

{

“accountId”:”7234234565”,

“priority”:”3”

}

],

“partyId”:”23654283”

}

]

}

}

**RESPONSE:**

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

“cardMessage”: {

“messageContext”: { <see messageContext.xsd>

},

“cardList”: [

{

“cardId”: “87923423467qq”,

“cardType”: ”Debit”,

“cardSubType”: “NFL”,

“linkedAccountList”:

[

{

“accountId”:”283746927364”,

“priority”:”1”

},

{

“accountId”:”5qw387987”,

“priority”:”2”

},

{

“accountId”:”7234234565”,

“priority”:”3”

}

],

“partyId”:”23654283”

}

]

}

}

### REST-JSON DELETE Card example

Note: This example deletes a Debit Card

**Required**: messageContext, at least one valid filter in cardFilter.

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: >=4.3.0

**PUT h**ttps://api.datasource.com/cardmessage

{

“cardMessage”:{

“messageContext”: { <see messageContext.xsd>

},

“cardList”: [

{

“cardId”: “87923423467qq”,

“cardType”: ”Debit”,

“cardSubType”: “NFL”,

“linkedAccountList”:

[

{

“accountId”:”283746927364”,

“priority”:”1”

},

{

“accountId”:”5qw387987”,

“priority”:”2”

},

{

“accountId”:”7234234565”,

“priority”:”3”

}

],

“partyId”:”23654283”

}

]

}

}

**RESPONSE**:

RESPONSE:

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{“cardMessage”: {

"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"

}

},

},

“cardList”: [

{

“cardId”: “87923423467qq”,

“cardType”:”Debit”,

“cardSubType”: “NFL”,

“linkedAccountList”:

[

{

“accountId”:”283746927364”,

“priority”:”1”

},

{

“accountId”:”5qw387987”,

“priority”:”2”

},

{

“accountId”:”7234234565”,

“priority”:”3”

}

],

“partyId”:”23654283”

}

]

}

}

### REST-JSON ACTIVATE Card example

Note: this is an example of the data elements involved in activating a card.

**Required**: messageContext, at least one valid filter in cardFilter, the specific id of the card to be updated in the message.

To activate a card, perform a card update, setting the cardStatus to Active and optionally specifying an activation date.

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

**PUT h**ttps://api.datasource.com/cardmessage

{

“cardMessage”:{

“messageContext”: { <see messageContext.xsd>

},

“cardFilter”:{

“cardIdList”:[

”87923423467qq”

]

},

“cardList”: [

{

“cardId”: “87923423467qq”,

“cardStatus”: ”Active”,

“activationDateTime”: “2013-03-03”

}

]

}

}

RESPONSE:

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{“cardMessage”: {

"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"

}

},

},

},

}

### REST-JSON BLOCK Card example

Note: this is an example of the data elements involved in blocking a card, or handling a lost, stolen or damaged card.

**Required**: messageContext, at least one valid filter in cardFilter, the specific id of the card to be updated in the message.

To block a card, set the cardStatus to Blocked, and optionally specify a block reason and blocked date.

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

**PUT h**ttps://api.datasource.com/cardmessage

{

“cardMessage”:{

“messageContext”: { <see messageContext.xsd>

},

“cardFilter”:{

“cardIdList”:[

”87923423467qq”

]

},

“cardList”: [

{

“cardId”: “87923423467qq”,

“cardStatus”: ”Blocked”,

“blockedReason”: “Lost card”,

“blockedDateTime”: “2013-03-03”

}

]

}

}

RESPONSE:

Headers:

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{“cardMessage”: {

"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"

}

},

},

“cardList”: [

{

“cardId”: “87923423467qq”,

“cardStatus”: ”Blocked”,

“blockedReason”: “Lost card”,

“blockedDateTime”: “2013-03-03”

}

]

}

}

# 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.