

card Data and Services 3.2

# 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\_3.2\_RFC\_Active

Previous Version: **Assembla**.com. Files Tag = CUFX\_3.1\_RFC\_Archive

# 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 |

# 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](#_Toc358361369)

[Authors and Change Log 1](#_Toc358361370)

[Overview of Specification 1](#_Toc358361371)

[Any known Errors in the document 2](#_Toc358361372)

[Table of Contents 2](#_Toc358361373)

[Document Conventions 2](#_Toc358361374)

[Definitions related to the specification 2](#_Toc358361375)

[Data Elements 2](#_Toc358361376)

[Filters used when accessing the card data 3](#_Toc358361377)

[card Data attributes 3](#_Toc358361378)

[CARD Services 3](#_Toc358361379)

[Overview 3](#_Toc358361380)

[Card Resource based create, read, update, delete services 4](#_Toc358361381)

[REST-JSON CREATE Card example 4](#_Toc358361382)

[REST-JSON READ Card example 5](#_Toc358361383)

[REST-JSON UPDATE Card example 6](#_Toc358361384)

[REST-JSON DELETE Card example 7](#_Toc358361385)

[REST-JSON ACTIVATE Card example 8](#_Toc358361386)

[REST-JSON BLOCK Card example 9](#_Toc358361387)

[General Error handling For All Services 9](#_Toc358361388)

[**Bibliography** 9](#_Toc358361389)

# 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.

# 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/card/ |
| 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, delete) * cufx:cardList (for create, update) |
| Outputs | cufx:cardList |
| Return Values | cufx: Error |
| 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** |

### 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: >=3.0.0

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

{

“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

{

“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: >=3.0.0

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

{

“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

{

“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: >=3.0.0

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

{

“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

{

“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: >=3.0.0

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

{

“cardMessage”:{

“messageContext”: { <see messageContext.xsd>

},

“cardFilter”:{

“cardIdList”:[

”87923423467qq”

]

}

}

}

**RESPONSE**:

Headers:

Status Code: 200 Ok

### 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: >=3.0.0

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

{

“cardMessage”:{

“messageContext”: { <see messageContext.xsd>

},

“cardFilter”:{

“cardIdList”:[

”87923423467qq”

]

},

“cardList”: [

{

“cardId”: “87923423467qq”,

“cardStatus”: ”Active”,

“activationDateTime”: “2013-03-03”

}

]

}

}

**RESPONSE:**

Headers:

Status Code: 200 Ok

### 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: >=3.0.0

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

{

“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

# 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.