

Validation Data and Services 5.0

# 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 |
| 0.0.1 |  | * Initial Creation |
| 0.0.2 |  | * <description> |
| 0.0.3 |  | * Added missing data. Filled in field information |
| 0.0.5 |  | * Added Examples. Changed score to creditScore to prepare for merging with lending XSD. |
| 0.0.6 |  | * Removed old security artifacts (MAC=) |
| 0.0.7 |  | * Changed to XSD Format |
| 0.0.8 |  | * Cleaned up some formatting and typo’s. corrected numbering convention |
| 0.0.9 |  | * Added MessageContext references. |
| 0.0.10 |  | * Updated reference naming |
| 0.0.11 |  | * Service name tweaks for CRUD approach |
| 0.0.12 |  | * Updated Error section to reference Error Mapping document * Update url’s to remove FiId |
| 0.0.13 |  | * Minor naming corrections to ID question services |
| 0.0.14 |  | * Delete author list, standardize version at top of doc. * Added data element section * Added messageContext where missing in messages and also got rid of references to fiUserId. * Change name of service to SimpleValidaitonRequest * Converted to examples in the bottom rather than specs. * Changed formatting of examples to provide * Create new XSD for creditReportRequest to be able to request multiple credit reports in one request. * Updated creditReport return to list multiple credit reports. * Add taxId and partyId to creditReport examples. * Converted Credit Reporting put to a post (it’s creating in back end system) * Change service names to make more CRUD centric * Convert all attributes to elements in XSD * Added identificationResponse resource * Convert to use general error message paragraph. |
| 0.0.15 |  | * Update Overview of Specification |
| 0.0.16 |  | * Update examples to reflect current state of the XSD’s. * Added ability to read and remove previously pulled credit report requests from data core. |
| 3.0 | **11/25/2013** | * Versioning and format change with release CUFX 3.0 * Switch to use X-HTTP-METHOD-OVERRIDE standard rather than subMethod non-Standard method for overriding request types. * Created a simpleValidationRequestMessage or creditReportMessage wrapper for every message to increase ability for infrastructure to serialize the data |
| 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,   Created SimpleValidationRequestFilter, added to SimpleValidationRequestMessage. |
| 4.1 | **12/10/2018** | * Updated to release 4.1, ValidationStatus - Replaced xmlns:xsd with xmlns:xs for consistency with the rest of standard. |
| 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 | **10/14/2020** | * Updated to release 4.4 |
| 4.5 | **04/02/2021** | * Updated to release 4.5 |
| 5.0 | **06/07/2021** | * Updated to release 5.0, SimpleValidationRequestFilter - Update PartyIdList to reference Common.xsd. |

# Overview of Specification

This specification defines the features for validating a potential member’s identity and the risk they might pose to the financial institution.

This is divided into three categories.

1. Simple pass fail validations. Information collected can be passed to any number of services which can then return a status indicating if the potential member is acceptable.
2. Services for managing credit reports.
3. Services for determining identity via out of wallet questions.

# Any know Errors in the document

|  |  |
| --- | --- |
| **Error Description** | Status of Error |
| REST-JSON READ IDENTIFICATION REQUEST-RESPONSE schema and example needs to be updated to align with the new call pattern for CUFX release 4.0. We anticipate a new Questions and Answers Message structure will be developed. The current example is functional, but is considered in containment pending the update. | Pending update with release 4.1. |

Table of Contents

[Document Status 1](#_Toc68102421)

[Change Log 1](#_Toc68102422)

[Overview of Specification 2](#_Toc68102423)

[Any know Errors in the document 2](#_Toc68102424)

[Document Conventions 3](#_Toc68102425)

[CUFX API and Documentation Support 4](#_Toc68102426)

[Release 4.0 Global Update Notes 4](#_Toc68102427)

[Release 4.4 Global Update Notes 4](#_Toc68102428)

[Definitions related to the specification 5](#_Toc68102429)

[Concepts 5](#_Toc68102430)

[Data Elements 5](#_Toc68102431)

[Filters used when accessing the card data 5](#_Toc68102432)

[Validation services Data attributes 5](#_Toc68102433)

[Services 6](#_Toc68102434)

[simpleValidationRequest 6](#_Toc68102435)

[Overview 6](#_Toc68102436)

[Authenticaton 6](#_Toc68102437)

[simpleValidationRequest Details 6](#_Toc68102438)

[REST-JSON READ REQUEST-RESPONSE Example 7](#_Toc68102439)

[CreditReporting 8](#_Toc68102440)

[Overview 8](#_Toc68102441)

[Authenticaton 8](#_Toc68102442)

[creditReporting Request Details 8](#_Toc68102443)

[Identification 13](#_Toc68102444)

[Identification Use Case 13](#_Toc68102445)

[Overview 14](#_Toc68102446)

[Authentication 14](#_Toc68102447)

[Identification Details 14](#_Toc68102448)

[Identification Answer Details 15](#_Toc68102449)

[REST-JSON CREATE IdentificaIton Answer REQUEST-RESPONSE 15](#_Toc68102450)

[General Error handling For All Services 16](#_Toc68102451)

[Bibliography 16](#_Toc68102452)

# 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.

# Release 5.0 Global Update Notes

CUFX Release 5.0 Moved PartyId and PartyIdList into the Common.xsd. Namespaces references to Party were removed if there were no other references to Party in the specification. PartyId and PartyIdList type references were updated.

# Definitions related to the specification

provides Risk assessment of New Members

DATA PROVIDER

Any Financial Institution, Core Banking Provider, or Online Banking Provider from which financial data is requested and returned using this specification.

# Concepts

Helping established identity and establish the risk posed by a new member

1. Simple Pass/fail
2. Full blown – detail service
3. Out of wallet questions

Service provider should specify if they’re providing the simple validation service, the full service or out of wallet or some combination. If it’s simple validation they should provide a list of the validations.

# 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.

## Validation services Data attributes

All CUFX fields related to a validation are defined in

* SimpleValidationRequest.xsd
* CreditReportRequest.xsd
* CreditReport.xsd
* CreditReportFIlter.xsd
* QuestionList.xsd
* Answer.xsd
* ValidationStatus.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

## simpleValidationRequest

### Overview

|  |  |
| --- | --- |
| Definition | Provides simple pass/fail response for the requested validation type. |
| Overview of Capabilities | Verify party information or qualifications. This may be for example checking against the OFAC list, retrieving Credit Bureau etc. and returning simple pass/fail response. Multiple services may implement this interface. |
| Dependencies | *Security Services* for authentication and security. *Party Services* for party data definition. *Error* specifications. |
| CUFX REST LINK | https://api.serviceprovider.com/simplevalidationrequestmessage |
| CUFX WSDL LINK |  |

### Authenticaton

The service provider will use a mechanism from the security services specification to authenticate the application.

### simpleValidationRequest Details

|  |  |
| --- | --- |
| Inputs | cufx:simpleValidationRequestMessage (which includes)   * [cufx:messageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * cufx:simpleValidationRequestFilter * cufx:simpleValidationRequestList |
| Outputs | cufx:simpleValidationRequestMessage (which includes)   * [cufx:messageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * cufx:simpleValidationRequestList |
| Return Values | cufx:simpleValidationRequestMessage (which includes)   * [cufx:messageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html)   + statusList |
| Service Specific Error Conditions | If the service provider requires a field in applicant that is not provided, the service provider should include the name of all missing required fields in the error. |
| Side Effects | None |
| Dependencies | Security Services for authentication and security.  Member Services for party data definition |
| CUFX REST LINK | https://api.serviceprovider.com/simplevalidationrequestmessage |
| CUFX WSDL LINK |  |
| Testing Procedures for Certification | Access the CUFX test service with a test data set that should pass and verify that it passed. Access the CUFX test service with a test data set that should fail and verify that it failed. Access the CUFX test service with a test data set that should return an error and verify that it returns an error and all missing fields are returned in the error object. |
| Derivative Services | None. |
| Fields Used | Message Headers : See security services  messageContext: See messageContext.xsd  Attributes: simpleValidationRequest : See simpleValidationRequest.xsd |

### REST-JSON READ REQUEST-RESPONSE Example

Consumers submit validation request to the service providers validation 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)*)*

X-API-Version: >=5.0.0

**POST** https://api.serviceprovider.com/simplevalidationrequestmessage

{

“simpleValidationRequestMessage”:{

“messageContext”: { <see messageContext.xsd>

},

“simpleValidationRequestList”:{

“simpleValidationRequest”:{

       “validationType”: “*OFAC*”,

       “applicant”: { /\*---See party data specification --\*/}

}

}

}

}

The Service Provider responds to Validate requests with a JSON-formatted message: See the error object for potential errors.

**RESPONSE:**

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

**Payload:**

{

"simpleValidationRequestMessage": {

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

"messageContext": {

"cufxVersion": "4.0.0",

"requestId": "requestId1",

"vendorId": "vendorId1",

"appId": "appId1",

"userList": {

"user": {

"userId": "userId1",

}

},

"statusList": {

"status": {

"statusType": "Success"

}

},

}

},

"simpleValidationRequestList": {

"simpleValidationRequest": {

"validationType": "validationType1",

"applicant": {

"applicantId": "applicantId1",

}

},

"relationshipId": "relationshipId1",

"partyId": "partyId1",

"party": {

"id": "id1",

}

}

},

"simpleValidationStatus": "Pass",

}

}

}

}

}

## CreditReporting

### Overview

|  |  |
| --- | --- |
| Definition | Provides detailed credit report services. |
| Overview of Capabilities | Retrieves detailed credit report(s) from service provider. Provide a credit report to a data provider. |
| Dependencies | Security Services and party Data specifications, messageContext specifications, and Error specifications. |
| CUFX REST LINK | https://api.serviceprovider.com/creditreportingrequestmessage |
| CUFX WSDL LINK |  |

### Authenticaton

The service provider will use a mechanism from the security services specification to authenticate the application.

### creditReporting Request Details

|  |  |
| --- | --- |
| INPUTS | cufx: creditReportMessage (which includes)   * cufx:MessageContext * cufx:CreditReportRequest * cufx:CreditReport |
| Outputs | cufx:*CreditReport* |
| Return Values | Cufx:Error |
| Side Effects | A credit report may be pulled |
| Dependencies | *Error handling, partyData and Security* Specifications |
| CUFX REST LINK | https://api.serviceprovider.com/creditreportingrequestmessage |
| CUFX WSDL LINK |  |
| Testing Procedures for Certification | Provide valid inputs to service and ensure that a credit report is returned. |
| Fields used | Message Headers : See security services  messageContext: See messageContext.xsd  Attributes: CreditReportRequest : See CreditReportRequest.xsd  Outputs: CreditReport: See CreditReport.xsd |

#### REST-JSON READ REQUEST-RESPONSE (retrieves credit Bureau(S))

The CreditReportRequest can take more than one applicant and return one or more credit bureau reports for each applicant depending upon the request. Consumers submit creditReportRequest:GET to the service providers creditReporting 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: >=5.0.0

**POST** https://api.serviceprovider.com/creditreportrequestmessage

{

“creditReportMessage”:{

“messageContext”: { <see messageContext.xsd>

},

“creditReportRequest”:{

“applicantList”: [

“applicant”:{

/\*---See partyData specification --\*/}

}

],

“maxReportAgeInDays”: 90,

“reportTypeList”:[

“Experian CreditProfile”,

“TransUnion EmployementReport”

]

}

}

}

The Service Provider responds to creditReporting requests with a JSON-formatted message:

**RESPONSE:**

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

**Payload:**

{

“creditReportMessage”:{

“messageContext”: { <see messageContext.xsd>

},

“creditReportList”: [

{“creditReportId”:”53434135”,

“creditSource”:”Experian”,

“reportType”:”Experian CreditProfile”,

“scoreType”:”FICO”,

“taxId”:”123456789”,

“creditScore”: 793,

“reportData”: “*NW#%###333...”* /\*--- this could be XML,

HTML, text, base64-encoded

binary report data ---\*/

},

{“creditSource”:”TransUnion”,

“reportType”:” TransUnion EmployementReport”,

“taxId”:”123456789”,

“reportData”: “*<xml...>”*

}

]

}

}

#### creditReporting CREATE Details (Places credit Bureau(S) in data system)

|  |  |
| --- | --- |
| INPUTS | cufx: creditReportMessage (which includes)   * cufx:MessageContext * cufx:CreditReportRequest * cufx:CreditReport |
| Outputs | *(none)* |
| Return Values | See Error Handling Specification |
| Side Effects | A credit report is stored |
| Dependencies | *Error handling, partyData and Security* Specifications |
| CUFX REST LINK | <https://api.dataprovider.com/creditReport>message |
| CUFX WSDL LINK |  |
| Testing Procedures for Certification | Provide credit report data and ensure that the data is stored. |
| Fields used | Message Headers : See security services  messageContext: See messageContext.xsd  Attributes: reditReport : See CreditReport.xsd |

#### REST-JSON CREATE REQUEST-RESPONSE (Places credit Bureau(S) in data system)

Client app submits creditReport create request to the data providers creditReport url. For all CUFX RESTful-JSON requests, the client must set these headers:

**REQUEST:**

Accept: application/json

Accept-Charset: utf-8

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

X-API-Version: >=5.0.0

**POST** https://api.dataprovider.com/creditreportmessage

{

“creditReportMessage”:{

“messageContext”: { <see messageContext.xsd>

},

“creditReportList”: [

{“creditSource”:”Experian”,

“reportType”:”Experian CreditProfile”,

“scoreType”:”FICO”,

“creditScore”: 793,

“partyId”:”1534834675634”,

“taxId”:”123456789”,

“reportData”: “*NW#%###333...”* /\*--- this could be XML,

HTML, text, base64-encoded

binary report data ---\*/

},

{“creditSource”:”TransUnion”,

“reportType”:” TransUnion EmployementReport”,

“partyId”:”1534834675634”,

“taxId”:”123456789”,

“reportData”: “*<xml...>”*

}

]

}

}

The data system responds to the creditReport create with a JSON-formatted message or an error message:

**RESPONSE:**

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

{

“creditReportMessage”:{

“messageContext”: { <see messageContext.xsd>

},

“creditReportList”: [

{“creditReportId”:”235318-153485634”

“creditSource”:”Experian”,

“reportType”:”Experian CreditProfile”,

“scoreType”:”FICO”,

“creditScore”: 793,

“partyId”:”1534834675634”,

“taxId”:”123456789”,

“reportData”: “*NW#%###333...”* /\*--- this could be XML,

HTML, text, base64-encoded

binary report data ---\*/

},

{“creditReportId”:”3518743483-ab385”

“creditSource”:”TransUnion”,

“reportType”:” TransUnion EmployementReport”,

“partyId”:”1534834675634”,

“taxId”:”123456789”,

“reportData”: “*<xml...>”*

}

]

}

}

#### REST-JSON READ a PReviously stored Credit Bureau PulL from the data system

Client app requests a previously stored credit bureau from the data system. For all CUFX RESTful-JSON requests, the client must set these headers:

**REQUEST:**

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: >=5.0.0

**POST** <https://api.dataprovider.com/creditreport>message

{

“creditReportMessage”:{

“messageContext”: { <see messageContext.xsd>

},

“creditReportFilter”: {

“creditReportIdList”:[

”235318-153485634”,

”3518743483-ab385”

]

}

}

}

The data provider responds to creditReporting read with a JSON-formatted message or an error message:

**RESPONSE:**

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

{

“creditReportMessage”:{

“messageContext”: { <see messageContext.xsd>

},

“creditReportList”: [

{“creditReportId”:”235318-153485634”

“creditSource”:”Experian”,

“reportType”:”Experian CreditProfile”,

“scoreType”:”FICO”,

“creditScore”: 793,

“partyId”:”1534834675634”,

“taxId”:”123456789”,

“reportData”: “*NW#%###333...”* /\*--- this could be XML,

HTML, text, base64-encoded

binary report data ---\*/

},

{“creditReportId”:”3518743483-ab385”

“creditSource”:”TransUnion”,

“reportType”:” TransUnion EmployementReport”,

“partyId”:”1534834675634”,

“taxId”:”123456789”,

“reportData”: “*<xml...>”*

}

]

}

}

#### REST-JSON DELETE a PReviously stored Credit Bureau PulL from the data system

Client app deletes a previously stored credit bureau from the data system. For all CUFX RESTful-JSON requests, the client must set these headers:

**REQUEST:**

Accept: application/json

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: >=5.0.0

**PUT** <https://api.dataprovider.com/creditreporting>message

{

“creditReportMessage”:{

“messageContext”: { <see messageContext.xsd>

},

“creditReportFilter”: {

“creditReportIdList”:[

”235318-153485634”,

”3518743483-ab385”

]

}

}

}

The data provider responds to creditReporting delete with a JSON-formatted message or an error message:

**RESPONSE:**

**Headers:**

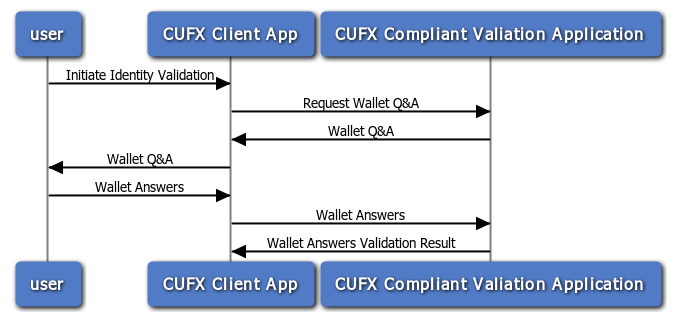
Status Code: 200 Ok

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

Content-Language: en-us

## Identification

### Identification Use Case



### Overview

|  |  |
| --- | --- |
| Definition | Out-of-wallet questions to confirm identity |
| Overview of Capabilities | Retrieve out-of-wallet questions with multiple-choice answers, and validates answers chosen. |
| Dependencies | *Security Services*, messageContext, *partyData* specifications. |
| CUFX REST LINK | https://api.serviceprovider.com/identification |
| CUFX WSDL LINK |  |

### Authentication

The typical use case is to provide user credentials with the <service-group> request. After obtaining a valid Session Key, the client software will provide this Key with all subsequent requests. See Security Services for more detail.

### Identification Details

|  |  |
| --- | --- |
| INPUTS | cufx:MessageContext  cufx:party |
| Outputs | [cufx:QuestionList](file:///C:\Users\brianl.STCU\Downloads\QuestionList.html) |
| Return Values | See Error Handling Specification |
| Dependencies | *Security Services* and *partyData* specifications. |
| CUFX REST LINK | https://api.serviceprovider.com/identificationquestionlist |
| CUFX WSDL LINK |  |
| Testing Procedures for Certification | Submit partyData and ensure that out-of-wallet data is returned in the language requested in the Accept-Language header. |
| Fields Used | Message Headers : See security services  messageContext: See messageContext.xsd  Attributes: party : See party.xsd  Outputs: questionlist: See QuestionList.xsd |

#### REST-JSON READ Identification REQUEST-RESPONSE

Client applications submit identification request which returns a question list from the service providers identification url. For all CUFX RESTful-JSON requests, the client must set these headers:

**REQUEST:**

**<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: >=5.0.0

**POST** https://api.serviceprovider.com/identificationQuestionList

{

“messageContext”: { <see messageContext.xsd>

},

“party”: { /\*---See party Data specification --\*/}

}

The Service Provider responds to identification question list request with a JSON-formatted message:

**RESPONSE:**

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

**Payload:**

{

“questionList”: [

{

“questionId” : “12353435534”,

“questionText” : “What was your car payment on your 1973 GREMLIN?”,

“choiceList” : [

{

“choiceId” : “A”,

“choiceText” : “$200-$299”

},

{

“choiceId” : “B”,

“choiceText” : “$300-$399”

},

{

“choiceId” : “C”,

“choiceText” : “$400-$499”

},

{

“choiceId” : “D”,

“choiceText” : “$500-$599”

},

{

“choiceId” : “E”,

“choiceText” : “None of the above”

}

]

}

]

}

### Identification Answer Details

|  |  |
| --- | --- |
| INPUTS | cufx:MessageContext  cufx:Answer |
| Outputs | [cufx:ValidationStatus](file:///C:\Users\brianl.STCU\Downloads\QuestionList.html) |
| Return Values | See Error Handling Specification |
| Dependencies | *Security Services, messageContext* and *partyData* specifications. |
| CUFX REST LINK | https://api.serviceprovider.com/identificationanswer |
| CUFX WSDL LINK |  |
| Testing Procedures for Certification | Submit answer data and ensure that out-of-wallet data is returned in the language requested in the Accept-Language header. |
| Fields Used | Message Headers : See security services  messageContext: See messageContext.xsd  Attributes: answer : See Answer.xsd  Output: validationStatus: See validationStatus.xsd |

### REST-JSON CREATE IdentificaIton Answer REQUEST-RESPONSE

Client application submits identification answer request to the service providers identification 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)*)*

X-API-Version: >=5.0.0

**POST** https://api.serviceprovider.com/identificationanswer

{

“messageContext”: { <see messageContext.xsd>

},

“answerList” : [ {

“party”:{ <See party services for details> },

“questionId” : “12353435534”,

“choiceList” : [

{

“choiceId” : “A”,

“choiceText” : “$200-$299”

},

{

“choiceId” : “B”,

“choiceText” : “$300-$399”

},

{

“choiceId” : “C”,

“choiceText” : “$400-$499”

},

{

“choiceId” : “D”,

“choiceText” : “$500-$599”

},

{

“choiceId” : “E”,

“choiceText” : “None of the above”

}

],

“answerChoice” : {

“choiceId” : “D”,

“choiceText” : “$500-$599”

}

}

]

}

The Service Provider responds to identification answer requests with a JSON-formatted message:

**RESPONSE:**

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

**Payload:**

{

{“validationStatus”:“Pass”}

}

# 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.