

LOAN Data Model 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 |
| 0.0.1 |  | * Initial Creation |
| 0.0.2 |  | * Changed services to use a more CRUD like approach |
| 0.0.3 |  | * Added the loanFilter |
| 0.0.5 |  | * Updated the Overview of Specification and Change Log |
| 0.0.6 |  | * Changed the account id reservation process |
| 0.0.7 |  | * Update id lists to make the consistent with XSD * Update TOC to reduce space * Updated graphic for reserving next loan ID * Found bad links in file and fixed * Changed accountId filter examples to relationshipId example since accountId and LoanId are the same thing. |
| 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 loanMessage or loanDisbursementMessage wrapper for every message to increase ability for infrastructure to serialize the data * Versioning and format change with release CUFX 3.0 |
| 3.0 | **12/13/2013** | * 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, Removed loanAccountStatus, loanAccountSubStatus, loanAccountStatusList and restructed AccountStatus to choice of Deposit,Loan, and Investment. Consolidated to Acccount.xsd. CreditLimitIncreaseRequestList - added min max occurs to elements. |
| 4.1 | **12/10/2018** | * Update 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, Account - Deprecated type. Use accountType in accountBase. Documented meta usage. * Added interestPostingFrequency, withholdingsInfoList, overDraftProtectionAccountId, overDraftProtectionAccountType, overDraftProtectionAccountSubType. * Added enums to accountType for Installment, CommercialLoan, CommercialLineOfCredit, GeneralLedger. OtherLoan. * DeprecatedAccountAccountTypeList and AccountAccountTypePair. * Added Type AccountIdentificationList, AccountIdentification, AccountSubType * Added enums ChargedOff, ChargeOffClosed to DepositAccountStatus" * Account Filter - includeNotesFlag type to common:IncludeNotesFlag. Added accountIdentificationList * Loan - Documentation updates. Deprecation statementCode, statementGroup, statementDate * Added numberOfPaymentsRemaining, otherLoanAccountCategory. * Added fromAccountType and fromAccountSubType to AutoPaymentOption. * Added paymentChangeDateTime, rateReviewDays, paymentReviewDays, rateChangeFrequencyType, rateChangeFrequencyDuration, paymentChangeFrequencyType, paymentChangeFrequencyDuration to LoanInterestRateDetail. * Updated maturityDate to use common MaturityDate. * Updated EcoaGroupType, added enum Deceased. * Added LoanAccountCategory enums Installment, Commercial, CommercialLineOfCredit, Mortgage, OtherLoan. * Added enum AmortizationTransfer to LoanPaymentMethodType * Added customData to LoanParty * LoanDisbursement - Added toAccountSubType to LoanDisbursementDepositToAccount |
| 4.5 | **04/02/2021** | * Updated to release 4.5 |

Overview of Specification

The Loan specification describes the services used as part of a new membership application process, for creating a loan, funding the loan, and disbursing the funds.

Any know Errors in the document

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

Table of Contents

[Document Conventions 3](#_Toc68100486)

[CUFX API and Documentation Support 3](#_Toc68100487)

[Release 4.0 Global Update Notes 4](#_Toc68100488)

[Release 4.4 Global Update Notes 4](#_Toc68100489)

[Definitions related to the specification 4](#_Toc68100490)

[High level use cases 5](#_Toc68100491)

[Use Case 1: Create a loan For new member 5](#_Toc68100492)

[State Diagram 5](#_Toc68100493)

[Data Elements 7](#_Toc68100494)

[Loan Origination Data attributes 7](#_Toc68100495)

[Data Element: Loan 7](#_Toc68100496)

[Data Element: LoanDisbursmentList 7](#_Toc68100497)

[Data Element: LoanFilter 7](#_Toc68100498)

[Data Element: MessageContext 7](#_Toc68100499)

[LOAN Services 8](#_Toc68100500)

[Service Definitions 8](#_Toc68100501)

[Service Message: Read Loan 8](#_Toc68100502)

[Service Message: Read Next Account (loan) Id 9](#_Toc68100503)

[Service Message: Create Loan 10](#_Toc68100504)

[Service Message: Update Loan 12](#_Toc68100505)

[Service Message: Create Loan Disbursements 13](#_Toc68100506)

[Bibliography 14](#_Toc68100507)

# 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

System of record

Authoritative data source for information such as the loans for which this specification will be creating.

Loan

Examples of types of loans would be a Line of Credit, a Car loan, Home Equity Line of Credit, or a Credit Card. Mortgages are currently out of scope for this version of the new membership application.

Loan Disbursment

Once a loan is creates, there could be zero or more disbursements of the funds for this loan. Example of a disbursement would be a check disbursement or a deposit to account disbursement.

# High level use cases

## Use Case 1: Create a loan For new member

This use case assumes that the calling application has already:

* Approved the member for membership, and the account (with related parties, etc.) has already been created on the core system.
* If the member had any derogatory relationships at the institution (e.g., they charged off a loan) the approving system would already have taken this into account and denied the loan product.

1

Create Loan w/idType=”Reserve” API

Request the next loan Id (Optional)

CUFX New Member Application

2

Create Reserve id Response

Contains the next loanId or error

3

CreateLoan API

Contains the loan and funds it

CUFX Core Data Provider

4

Loan Creation Response (CreateLoanAPI) contains confirmation or error

3

Disburse loan funds (DisburseFunds API) loan disbursement information such as printing a check, transfers, cash

4

Disburse loan funds response (DisburseFunds API) contains confirmation or error

# State Diagram

The following diagram lists the potential states (nodes) and transitions (edges) for the loan origination workflow. There are three possible end states, Error, Not Eligible (meaning the member and one or more derogatory conditions on the system of record, and is now not eligible for the loan), and Success (meaning the loan was successfully created, and any funds were disbursed.



# Data Elements

## Loan Origination Data attributes

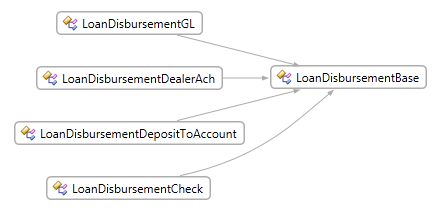
The following CUFX fields referenced in the services defined below are utilized for loan origination:

### Data Element: Loan

The Loan object, which the definition can be found in **Loan.xsd**, contains all the fields necessary for the creation of a loan, including the associated parties on the loan. It is important to note that the loan object contains a collection called the partyList. This list contains all parties associated to this loan, and their associated relationship. For example, a member could be the joint owner of the loan, and another member could be a co-signer. Both of these members would be parties listed in this partyList, with the appreciate relationship.

### Data Element: LoanDisbursmentList

The LoanDisbursmentList object, which the definition can be found **LoanDisbursement.xsd**, contains all the fields necessary for disbursing a loan. There are currently four types of disbursements that all inherit from a base LoanDisbursment object that contains the required fields used for all disbursements. The LoanDisbursementList also contains the loanId. The loanId used to tie back the loan disbursements to the original loan.



### Data Element: LoanFilter

The LoanFilter object, which the definition can be found in **LoanFilter.xsd**, contains fields for returning one or more loans that match a certain criteria. For example, if attempting to get all of the loans for a given relationship, you would create a loanFilter that contains the relationshipId within the relationshipIdList element of the loanFilter. This would then return all loans for that given relationship.

### Data Element: MessageContext

Like many of the other CUFX service, a MessageContext object must be passed for each request. Refer to the Security Services documentation for more information.

# LOAN Services

## Service Definitions

The loan data model and services support Create, Read, and Update operations against a Loan. Delete operations for a loan is not permitted, and instead it is recommend that setting the loan close date is the proper way to remove a loan from an account. The Create operations are supported for the LoanDisbursment object only.

### Service Message: Read Loan

When provided a loanFilter, the Read Loan service message will return a list of loans that match the filter. For example, the loanFilter when containing an relationshipId would return all loans associated to that relationship.

|  |  |
| --- | --- |
| INPUTS | cufx:loanMessage (which includes)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * [cufx:Loan](file:///\\files2\users\CMarjaniemi\Projects\CUFX\Generated%20HTML%20Docs\Loan.html)Filter (for read, update) * [cufx:Loan](file:///\\files2\users\CMarjaniemi\Projects\CUFX\Generated%20HTML%20Docs\Loan.html)List (for create, update, delete) |
| Outputs | cufx:loanMessage (which includes)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * [cufx:Loan](file:///\\files2\users\CMarjaniemi\Projects\CUFX\Generated%20HTML%20Docs\Loan.html)List |
| Return Values | cufx:loanMessage (which includes)   * cufx:MessageContext   + statusList |
| Side Effects | No data is being manipulated, only a lookup is being performed. |
| Dependencies | None. |
| CUFX REST LINK | https://api.dataprovider.com/loanmessage |

REST-JSON Example

The following example illustrates how to return all loans on relationshipId 12345 that are lines of credit or credit cards.

**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 h**ttps://api.dataprovider.com/loanmessage

{

“loanMessage”:{

"messageContext" :{ <SeeMessageContext.xsd>

},

"loanFilter" : {

"loanCategoryList" : [ "LineOfCredit", "CreditCard" ],

"relationshipIdList" : [ "12345" ]

}

}

}

### Service Message: Read Next Account (loan) Id

In some cases the new membership application will require an account ID (e.g. 1234-L13) for used in the documentation for the member to sign, prior to the creation of the loan on the core system. This is done by passing in a Loan that as the idtype set to “Reserved”.

|  |  |
| --- | --- |
| INPUTS | cufx:loanMessage (which includes)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * cufx:loanFilter (for read, update) * [cufx:loan](file:///\\files2\users\CMarjaniemi\Projects\CUFX\Generated%20HTML%20Docs\Loan.html)list(for create, update, delete) |
| Outputs | cufx:loanMessage (which includes)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * [cufx:loan](file:///\\files2\users\CMarjaniemi\Projects\CUFX\Generated%20HTML%20Docs\Loan.html)list |
| Return Values | cufx:loanMessage (which includes)   * cufx:MessageContext   + statusList |
| Side Effects | No data is being manipulated, only a lookup is being performed. |
| Dependencies | The account that this loan will belong to must already exist. |
| CUFX REST LINK | https://api.dataprovider.com/loanmessage |

REST-JSON Example

The following example illustrates how to get the next loan id on relationship 1234.

**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.dataprovider.com/loan](https://api.dataprovider.com/loan)message

{

“loanMessage”:{

"messageContext" : <See MessageContext.xsd>,

"loanList": {

"loan": { "idType" : “Reserved”,

“type” : “Loan”,

“relationshipId”:”1234”}

}

}

**RESPONSE:**

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

"loanMessage": {

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

"messageContext": {

"cufxVersion": "4.0.0",

"requestId": "requestId1",

"vendorId": "vendorId1",

},

"statusList": {

"status": {

"statusType": "Success"

}

},

"loanList": {

"loan": {

"accountId": "accountId1",

"type": "Checking",

}

}

}

}

}

}

### Service Message: Create Loan

The Create Loan service message will create the loan record on the host system, in a funded state. A subsequent call is made to disburse the funds for the loan (e.g. Check).

|  |  |
| --- | --- |
| INPUTS | cufx:loanMessage (which includes)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * cufx:loanFilter (for read, update) * [cufx:loan](file:///\\files2\users\CMarjaniemi\Projects\CUFX\Generated%20HTML%20Docs\Loan.html)list(for create, update, delete) |
| Outputs | cufx:loanMessage (which includes)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * [cufx:loan](file:///\\files2\users\CMarjaniemi\Projects\CUFX\Generated%20HTML%20Docs\Loan.html)list |
| Return Values | cufx:loanMessage (which includes)   * cufx:MessageContext   + statusList |
| Side Effects | Creates a new loan on the core system if successful. |
| Dependencies | The relationship that this loan will belong to must already exist. |
| CUFX REST LINK | https://api.dataprovider.com/loanmessage |

REST-JSON Example

**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. dataprovider.com/loanmessage

{

“loanMessage”:{

"messageContext" : {<See MessageContext.xsd>,

},

"loanList": {

"loan" : { <See Loan.xsd> }

}

}

**RESPONSE:**

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

"loanMessage": {

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

"messageContext": {

"cufxVersion": "4.5.0",

"requestId": "requestId1",

"vendorId": "vendorId1",

},

"statusList": {

"status": {

"statusType": "Success"

}

},

"loanList": {

"loan": {

"accountId": "accountId1",

"type": "Checking",

}

}

}

}

}

}

### Service Message: Update Loan

The Update Loan service message is used to update the loan record on the host system. A valid loanFilter must be passed in that indicate the loan or loans to be updated.

|  |  |
| --- | --- |
| INPUTS | cufx:loanMessage (which includes)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * cufx:loanFilter (for read, update) * [cufx:loan](file:///\\files2\users\CMarjaniemi\Projects\CUFX\Generated%20HTML%20Docs\Loan.html)list(for create, update, delete) |
| Outputs | cufx:loanMessage (which includes)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * [cufx:loan](file:///\\files2\users\CMarjaniemi\Projects\CUFX\Generated%20HTML%20Docs\Loan.html)list |
| Return Values | cufx:loanMessage (which includes)   * cufx:MessageContext   + statusList |
| Side Effects | Creates a new loan on the core system if successful. |
| Dependencies | The account that this loan will belong to must already exist. |
| CUFX REST LINK | https://api.dataprovider.com/loanmessage |

REST-JSON Example

**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 h**ttps://api.dataprovider.com/loanmessage

{

“loanMessage”:{

"messageContext" : <See MessageContext.xsd>,

"loanFilter" : { "accountIdList" : [ "1234" ] }

"loanList": {

"loan" : { <See Loan.xsd> }

}

}

}

**RESPONSE:**

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

"loanMessage": {

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

"messageContext": {

"cufxVersion": "4.5.0",

"requestId": "requestId1",

"vendorId": "vendorId1",

},

"statusList": {

"status": {

"statusType": "Success"

}

},

"loanList": {

"loan": {

"accountId": "accountId1",

"type": "Checking",

}

}

}

}

}

}

### Service Message: Create Loan Disbursements

The Create Loan Disbursements service message is used to disburse the proceeds of a loan, once the loan was successfully created. There can be multiple disbursements of varying type (e.g., Check disbursement, Account disbursement, etc.).

|  |  |
| --- | --- |
| INPUTS | cufx:loanDisbursementMessage (which includes)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * cufx:loanDisbursementFilter (for read, update) * [cufx:loanDisbursement](file:///\\files2\users\CMarjaniemi\Projects\CUFX\Generated%20HTML%20Docs\Loan.html)list(for create, update, delete) |
| Outputs | cufx:loanMessage (which includes)   * [cufx:MessageContext](file:///\\files2\users\CMarjaniemi\Projects\CUFX\MessageContext.html) * [cufx:loanDisbursement](file:///\\files2\users\CMarjaniemi\Projects\CUFX\Generated%20HTML%20Docs\Loan.html)list |
| Return Values | cufx: loanDisbursementMessage (which includes)   * cufx:MessageContext   + statusList |
| Side Effects | The funds will be transferred from the new loan to the required disbursements. |
| Dependencies | The loan must have already been created successfully via a call to CreateLoan. |
| CUFX REST LINK | https://api.dataprovider.com/loandisbursementmessage |

REST-JSON Example

**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.dataprovider.com/loandisbursementmessage

{

“loanDisbursementMessage”:{

"messageContext" : {<SeeMessageContext.xsd>,

},

" loanDisbursementList " : { <See LoanDisbursement.xsd> }

}

}

**RESPONSE:**

**Headers:**

Status Code: 200 Ok

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

Content-Language: en-us

Payload:

{

"loanDisbursementMessage": {

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

"messageContext": {

"cufxVersion": "4.5.0",

"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": {

"currentPage": "1",

"totalPages": "1",

"pageSize": "1",

"positionIndex": "positionIndex1",

"returnPage": "1",

"returnPositionIndex": "returnPositionIndex1",

"statusList": {

"status": {

"statusType": "Success"

},

},

"loanDisbursementList": {

"loanDisbursement": {

"loanDisbursementCheck": {

"loanDisbursementId": "loanDisbursementId1",

"amount": "1",

"description": "description1",

"comment": "comment1",

"payeeLines": "payeeLines1"

}

},

"loanId": "loanId1"

}

}

}

# Bibliography

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