Mobile Money API Specification 1.2.0 Disbursements

Document Summary

Official Document Number, Document Title and Version Number	Mobile Money API Specification 1.2.0 - Disbursements
Official Document Type	Non-binding Permanent Reference Document
Change Request Security Classification	Non-confidential

Document History

Document Version	API Release Version	Date	Brief Description of Change	Editor / Company
0.1	1.2.0-beta	Aug 2020	• Initial draft of document	GSMA
0.2	1.2.0	Sep 2020	• Implemented RFC 2020-3, RFC 2020-10, RFC 2020-13, RFC 2020-17, RFC 2020-18	GSMA

Other Information

Type	Description
Document Owner	Mobile Money API Working Group
Editor / Company	GSMA

Table of Contents

1 Introduction

The purpose of this document is to specify the endpoints, fields, objects, and enumerations for Disbursement Mobile Money APIs, which are a subset of the GSMA Mobile Money API Specification. The Disbursement Mobile Money APIs allow organisations to disburse funds to mobile money recipients.

For further reading, please refer to the following documents:

- **Mobile Money API Introduction**. Introduces the use and benefits of the Mobile Money API. Also provides a glossary of terms used by the Mobile Money API specifications.
- **Mobile Money API Fundamentals**. Specifies the design principles, behaviours, and error handling of the Mobile Money API.
- **Mobile Money API Master Specification**. Documents all Mobile Money API endpoints, fields, objects, and enumerations.

All documentation can be found on the GSMA Mobile Money API Developer Portal.

This document contains the following sections:

- API Endpoints
- Supporting Objects
- Enumerations
- API Sequence Diagrams

1.1 Intended Audience

Audience	Usage	Role
Mobile Money Providers	• To understand how to implement the Mobile Money API to allow organisations to disburse payments in singular and in bulk.	API Provider
NGOs	• To understand how to implement the Mobile Money API to provide aid payments to mobile money recipients.	API Consumer
Government Departments	• To understand how to implement the Mobile Money API to provide social credit payments to mobile money recipients.	API Consumer
Employers	• To understand how to implement the Mobile Money API to provide salary payments to mobile money recipients.	API Consumer

2 API Endpoints

API endpoint fields are described in this specification as follows:

- The field **name**.
- The field **type**.
- **Description** of the field.
- **Optionality** of the field, i.e. whether the field must be supplied. Optionality is identified as per follows:

→	Request optionality
←	Response optionality
O	Field is optional
M	Field is mandatory
C	Field is conditional
NA	Field does not need to be supplied. If supplied, it will be ignored.

- **Reference** where the field is an array and/or is defined by another object.
- **Validation** applied to the field, including enumeration, field length and use of regular expressions to validate format.

Please note that string fields have a default maximum length of 256 characters unless specified otherwise.

2.1 Transactions API

Individual disbursements can be created and viewed using transactions APIs. The following paths are permitted:

Operation	Path	Description
Create	*POST /transactions/type/{transactiontype}*	To be used for transaction creation when the provider's API Gateway requires that the transaction *type* be identified in the URL.
View	*GET /transactions/{transactionReference}*	To view a transaction.
Update	*PATCH /transactions/{transactionReference}*	To update the *transactionStatus* of a transaction.

For batches of disbursements, please refer to Bulk Disbursement.

2.1.1 Transaction UML Class Diagram

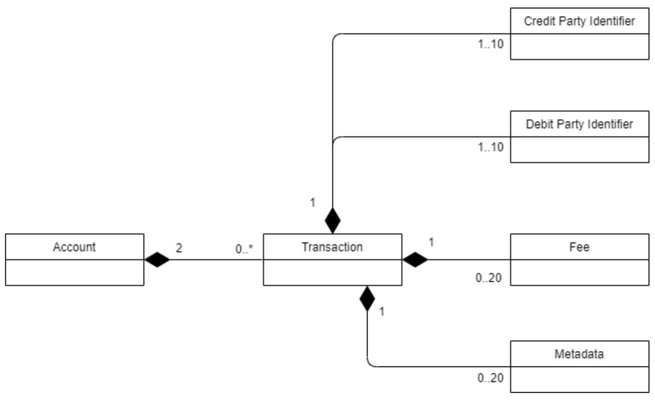

Figure 2-1 Transaction UML Class Diagram

2.1.2 Transaction Object Definition

Transaction Object					
Name	**Type**	**Description**		**Reference**	**Validation**
transactionReference	string	Unique reference for the transaction. This is returned in the response by API provider.	→NA ←M		
requestingOrganisationTransactionReference	string	A reference provided by the requesting organisation that is to be associated with the transaction.	→O ←O		

originalTransactionReference	string	For reversals and refunds, this field indicates the transaction which is the subject of the reversal.	→O ←O		
creditParty	array	A series of key/value pairs that enable the credit party to be identified. Keys include MSISDN and Wallet Identifier.	→C ←C	Account Identifiers	creditParty must be supplied if debitParty is omitted. If debitParty is supplied, then creditParty is optional.
debitParty	array	A collection of key/value pairs that enable the debit party to be identified. Keys include MSISDN and Wallet Identifier.	→C ←C	Account Identifiers	debitParty must be supplied if creditParty is omitted. If creditParty is supplied, then debitParty is optional.
type	string	The harmonised Transaction Type (not required if passed in the URL).	→M ←M		Enumeration = Transaction Types
subType	string	A non-harmonised sub-classification of the type of transaction. Values are not fixed, and usage will vary according to Provider.	→O ←O		
transactionStatus	string	Indicates the status of the transaction as stored by the API provider.	→NA ←M		
amount	string	The transaction amount.	→M ←M		Please refer to API Fundamentals document for amount validation rules.
currency	string	Currency of the transaction amount.	→M ←M		Enumeration = ISO Currency Codes
descriptionText	string	Free format text description of the transaction provided by the client. This can be provided as a reference for the receiver on a notification SMS and	→O ←O		

		on an account statement.			
fees	array	Allows the passing and/or returning of all fees pertaining to the transaction.	→O ←O	Fees Object	
geoCode	string	Indicates the geographic location from where the transaction was initiated.	→O ←O		
oneTimeCode	string	A one-time code that can be supplied in the request or can be generated in the response depending upon the use case. An authorisation code can be supplied in this field for requests that have been pre-authorised.	→O ←O		
requestingOrganisation	object	The originating organisation of the request.	→O ←O	Requesting Organisation	
servicingIdentity	string	The field is used to identify the servicing identity for transactions, e.g. till, POS ID, assistant ID.	→O ←O		
transactionReceipt	string	Transaction receipt number as notified to the parties. This may differ from the Transaction Reference.	→NA ←O		
creationDate	date-time	Date and time when the transaction was created by the API Provider.	→NA ←O		
modificationDate	date-time	Date and time when the transaction was modified by the API Provider.	→NA ←O		
requestDate	date-time	The date and time of the transaction request as supplied by the client.	→O ←O		
customData	string	A collection of key/value pairs that can be used for provider specific fields.	→O ←O	Custom Data Object	

metadata	array	A collection of key/value pairs. These can be used to populate additional properties that describe administrative information regarding the transaction.	→O ←O	Metadata	

2.2 Reversals API

The Reversals API is used to reverse, adjust, or refund a disbursement. The originating transaction reference must be provided in the path in order to identify the disbursement to be reversed. For a partial reversal, the amount needs to be supplied.

For viewing reversals and updating reversals, the Transactions API should be used

The supported path is *POST /transactions/{originalTransactionReference}/reversals*.

2.2.1 Reversal UML Class Diagram

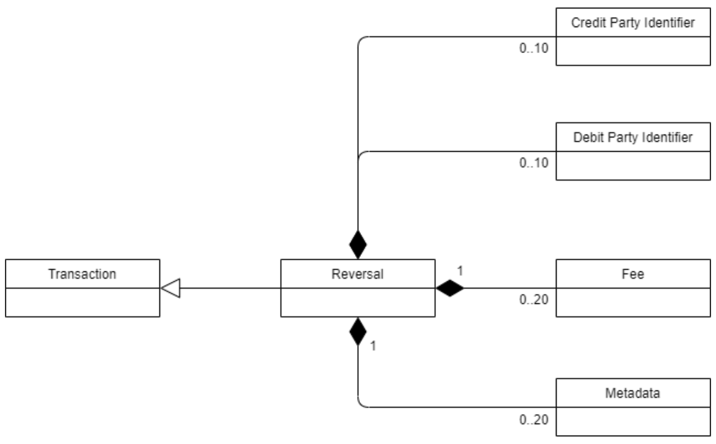

Figure 2-2 Reversal UML Class Diagram

2.2.2 Reversal Object Definition

Reversal Object					
Name	**Type**	**Description**		**Reference**	**Validation**
transactionReference	string	Unique reference for the transaction. This is returned in the response by API provider.	→NA ←M		
requestingOrganisationTransactionReference	string	A reference provided by the requesting organisation that is to be associated with the transaction.	→O ←O		
originalTransactionReference	string	For reversals and refunds, this field indicates the transaction which is the subject of the reversal.	→NA ←M		

creditParty	array	A series of key/value pairs that enable the credit party to be identified. Keys include MSISDN and Wallet Identifier.	→O ←O	Account Identifiers	
debitParty	array	A collection of key/value pairs that enable the debit party to be identified. Keys include MSISDN and Wallet Identifier.	→O ←O	Account Identifiers	
type	string	The harmonised Transaction Type.	→M ←M		Enumeration = Transaction Types Note that only Reversals and Refunds (adjustments) are supported.
subType	string	A non-harmonised sub-classification of the type of transaction. Values are not fixed, and usage will vary according to Provider.	→O ←O		
transaction Status	string	Indicates the status of the transaction as stored by the API provider.	→NA ←M		
amount	string	The transaction Amount.	→O ←O		Please refer to API Fundamentals document for amount validation rules.
currency	string	Currency of the transaction amount.	→O ←O		Enumeration = ISO Currency Codes.
description Text	string	Free format text description of the transaction provided by the client. This can be provided as a reference for the receiver on a notification SMS and on an account statement.	→O ←O		
fees	array	Allows the passing and/or returning of all fees pertaining to the transaction.	→O ←O	Fees Object	
geoCode	string	Indicates the geographic location from where the	→O ←O		

		transaction was initiated.			
requesting Organisation	object	The originating organisation of the request.	→O ←O	Requesting Organisation	
servicingIdentity	string	The field is used to identify the servicing identity for transactions, e.g. till, POS ID, assistant ID.	→O ←O		
transaction Receipt	string	Transaction receipt number as notified to the parties. This may differ from the Transaction Reference.	→NA ←O		
creationDate	date-time	Date and time when the transaction was created by the API Provider.	→NA ←O		
modificationDate	date-time	Date and time when the transaction was modified by the API Provider	→NA ←O		
requestDate	date-time	The date and time of the transaction request as supplied by the client.	→O ←O		
customData	string	A collection of key/value pairs that can be used for provider specific fields.	→O ←O	Custom Data Object	
metadata	array	A collection of key/value pairs. These can be used to populate additional properties that describe administrative information regarding the transaction.	→O ←O	Metadata	

2.3 Batch Transactions

The Mobile Money API allows organisations to submit, approve and view batches of disbursements. The following steps describe an end to end flow for processing batch disbursements. Two types of processing modes are supported:

- One shot processing without an approver.
- Maker/checker approval, i.e. transactions are not completed until approved via a separate API request.

The processing mode is determined by the financial service provider.

The individual APIs that are referenced in the steps below are fully documented in subsequent sub-sections.

2.3.1 Batch Transactions Workflow

2.3.1.1 One-Shot Batch Processing

Creating a Batch

1. Client submits the batch for processing via *POST /batchtransactions*.
2. The client will return the Request State object indicating whether a callback will be provided or polling is required.
3. The API provider will parse the batch in order to determine whether the transactions are capable of being processed.
4. Once parsing has completed, the API provider will set the batch status in the *batchTransactions* object to ‘**completed**’.

Verifying a Batch

5. The client will be able to retrieve the batch transaction object by invoking *GET /batchtransactions* using the object reference provided by the *requestState* object. Alternatively, if Callback is specified, the client will receive the representation of the *batchTransactions* object to their nominated URL set in the X-Callback-URL header.
6. If errors are indicated, i.e. some of the transactions failed parsing, the client is able to retrieve the errors via *GET /batchtransactions/rejections*. Successfully completed transactions can be viewed via *GET /batchtransactions/completions*.

2.3.1.2 Batch Processing with Maker/Checker

Creating a Batch

1. Client submits the batch for processing via *POST /batchtransactions*.
2. The client will return the *requestState* object indicating whether a callback will be provided or polling is required.
3. The API provider will parse the batch in order to determine whether the transactions are capable of being processed.

4. Once parsing has completed, the API provider will set the batch status in the *batchTransactions* object to ‘**created**’.

Verifying a Batch

5. The client will be able to retrieve the batch transaction object by invoking *GET /batchtransactions* using the object reference provided by the Request State object. Alternatively, if Callback is specified, the client will receive the representation of the batch transaction object to their nominated URL set in the X-Callback-URL header.
6. If errors are indicated, i.e. some of the transactions failed parsing, the client is able to retrieve the errors via *GET /batchtransactions/rejections*.

Approving a Batch

7. A client can approve the batch for posting by issuing a *PATCH /batchtransactions* in order to update the status to ‘**approved**’.
8. As per step 2, a *requestState* object will be returned indicating whether a callback will be provided or polling is required.
9. The API provider will then post the transactions in the batch considering any scheduling considerations.
10. Once posting is completed, the API provider will set the batch status in the *batchTransactions* object to ‘**completed**’.
11. The client will be able to retrieve the *batchTransactions* object by invoking GET /batchtransactions using the object reference provided by the *requestState* object. Alternatively, if Callback is specified, the client will receive the representation of the *batchTransactions* object to their nominated URL set in the X-Callback-URL header.
12. the client will also be able to retrieve a list of successful transaction completions */batchtransactions/completions* and transaction failures */batchtransactions/rejections*.

2.4 Batch Transactions API

This API enables organisations to submit and approve a batch of transactions. The API allows transactions of multiple types to be include in a single batch. The following paths are permitted:

- **Submit** a batch: *POST /batchtransactions*
- **Approve** a batch: *PATCH /bathtransactions/{batchID}*. The Batch Status needs to be set to ‘approved’.
- **View** details regarding batch processing: *GET /batchtransactions/{batchID}*

Only asynchronous mode is supported for the POST and PATCH methods. For the GET method, only synchronous mode is supported.

There is a limit of 999,999 transaction records per batch.

2.4.1 Batch Transaction Object UML Diagram

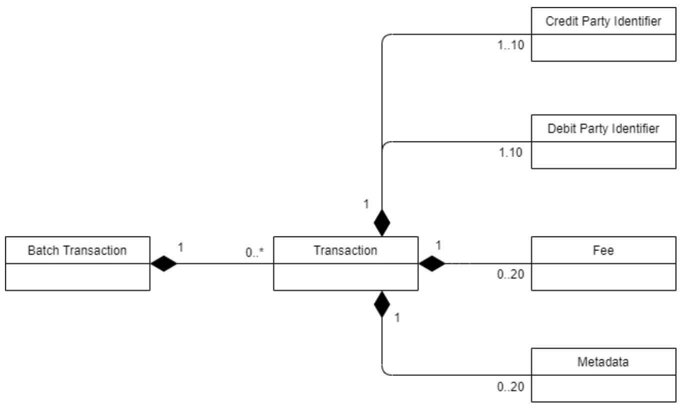

Figure 2-3 Batch Transaction UML Class Diagram

2.4.2 Batch Transaction Object Definition

Batch Transaction Object					
Name	**Type**	**Description**		**Reference**	**Validation**
batchID	string	Identifier for the Batch that is assigned by the API provider. This ID is used by the client on subsequent GET or PATCH methods.	→N/A ←M		
batchStatus	string	Indicates the status of the batch.	→O ←M		Enumeration = created, approved, completed
Transactions	array	Collection of Transactions that are to be processed. Note that the representation of each completed transaction can be retrieved via the '/completions API.	→M ←N/A	Transactions	
approvalDate	date-time	Indicates when the batch was approved as recorded by the API provider.	→NA ←M		
completionDate	date-time	Indicates when the batch was completed as recorded by the API provider.	→NA ←M		
batchTitle	string	Client-provided title for the batch.	→O ←O		
batchDescription	string	Client-provided description of the batch.	→O ←O		

processingFlag	boolean	Indicates whether the batch is currently undergoing processing by the API Provider.	→N/A ←O		
completedCount	integer	Indicates the number of records that have been successful completed.	→NA ←O		
parsingSuccessCount	integer	Indicates the number of records that have been parsed successfully.	→NA ←O		
rejectionCount	integer	Indicates the number of records that have been rejected, either during parsing or during final processing.	→NA ←O		
requestingOrganisation	object	The originating organisation of the request.	→O ←O	Requesting Organisation	
scheduledStartDate	date-time	If the batch has been scheduled, the expected start time is provided here.	→O ←O		
creationDate	date-time	Indicates when the batch was created as recorded by the API provider.	→NA ←O		
modificationDate	date-time	Indicates when the batch was modified as recorded by the API provider.	→NA ←O		
requestDate	date-time	The date and time of the batch request as supplied by the client.	→O ←O		
customData	string	A collection of key/value pairs that can be used for provider specific fields.	→O ←O	Custom Data Object	

2.5 Batch Rejections API

This API enables organisations to retrieve information on all transactions that have either failed parsing or have failed to be completed. Only the GET method is supported. The path is *batchtransactions/{batchID}/rejections*.

To filter the number of records returned, the following query strings can be used:

Parameter	Type	Format	Description

limit	integer	N/A	Supports pagination. If this is not supplied, then the server will apply a limit of 50 records returned for each request.
offset	integer	N/A	Supports pagination. This value will indicate the cursor position from where to retrieve the set of records. For example, a limit of 50 and offset of 10 will return records 11 to 60.
fromDateTime	string	date-time	Indicates the minimum creationDate for which records should be returned.
toDateTime	string	date-time	Indicates the maximum creationDate for which records should be returned.

Note: HTTP response headers are returned with each response indicating the total number of records available (X-Records-Available-Count) and total number of records returned (X-Records-Returned-Count).

2.5.1 Batch Rejection UML Class Diagram

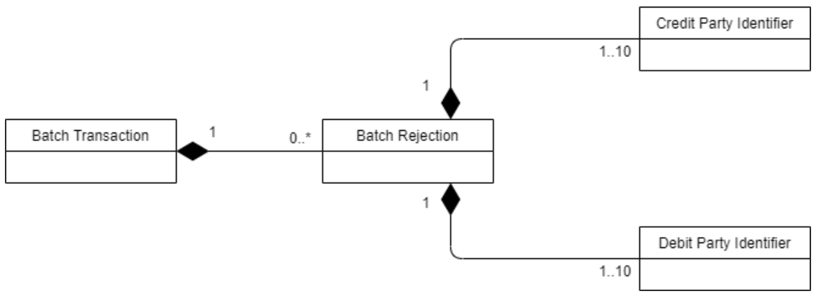

Figure 2-4 Batch Rejection UML Class Diagram

2.5.2 Batch Rejection Object Definition

Batch Rejection Object					
Name	**Type**	**Description**		**Reference**	**Validation**
transaction Reference	string	Transaction Reference as assigned by the API provider.	→N/A ←O		
requesting Organisatio nTransactio nReference	string	A reference provider by the requesting organisation that is to be associated with the transactions.	→N/A ←O		
creditParty	array	The credit party identifiers for the transaction as specific in the batch request.	→N/A ←M	Account Identifiers	

debitParty	array	The debit party identifiers for the transaction as specific in the batch request.	→N/A ←M	Account Identifiers	
rejectionReason	string	The reason for the transaction request as indicated by the API provider.	→N/A ←M		
rejectionDate	date-time	Date and time of the rejection.	→N/A ←M		
customData	string	A collection of key/value pairs that can be used for provider specific fields.	→O ←O	Custom Data Object	

2.6 Batch Completions API

This API lists all transactions that have successfully completed for a given batch. Only the GET method is supported. The path format is *batchtransactions/{batchID}/completions*.

To filter the number of records returned, the following query strings can be used:

Parameter	Type	Format	Description
limit	integer	N/A	Supports pagination. If this is not supplied, then the server will apply a limit of 50 records returned for each request.
offset	integer	N/A	Supports pagination. This value will indicate the cursor position from where to retrieve the set of records. For example, a limit of 50 and offset of 10 will return records 11 to 60.
fromDateTime	string	date-time	Indicates the minimum creationDate for which records should be returned.
toDateTime	string	date-time	Indicates the maximum creationDate for which records should be returned.

Note: HTTP response headers are returned with each response indicating the total number of records available (X-Records-Available-Count) and total number of records returned (X-Records-Returned-Count).

2.6.1 Batch Completion UML Class Diagram

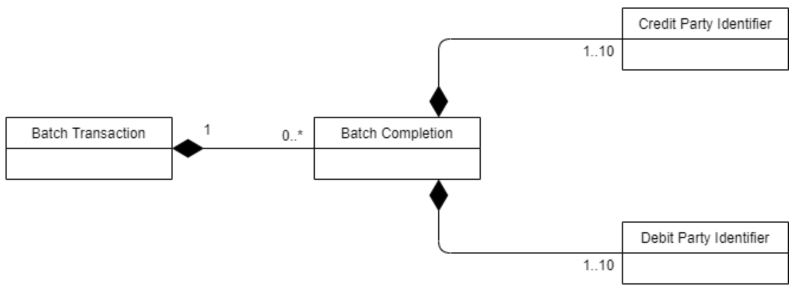

Figure 2-5 Batch Completion UML Class Diagram

2.6.2 Batch Completion Object Definition

Batch Completion Object					
Name	**Type**	**Description**		**Reference**	**Validation**
transactionReference	string	Transaction Reference as assigned by the API provider.	→N/A ←M		
requestingOrganisationTransactionReference	string	A reference provided by the requesting organisation that is to be associated with the transactions.	→N/A ←O		
creditParty	array	The credit party identifiers for the transaction as specified in the batch request.	→N/A ←M	Account Identifiers	
debitParty	array	The debit party identifiers for the transaction as specified in the batch request.	→N/A ←M	Account Identifiers	
completionDate	date-time	Date and time indicating when the transaction was completed.	→N/A ←M		
link	string	Provides a URL to the transaction resource.	→N/A ←M		
customData	string	A collection of key/value pairs that can be used for provider specific fields.	→O ←O	Custom Data Object	

2.7 Disbursement Accounts APIs

Using the mobile money APIs, disbursement organisations can:

- View transactions for their account.
- View their account balance.

2.7.1 Identifying a Disbursement Organisation Account

Two methods are provided for identifying a disbursement organisation account, the single identifier method, and the multiple identifiers method.

2.7.1.1 Single Identifier Method

In the scenario where one identifier suffices to uniquely identify an account, the following path is to be used: */accounts/{identifierType}/{identifier}*.

2.7.1.2 Multiple Identifiers Method

Where a single identifier is not sufficient to identify an account, the following path is to be used:
/accounts/{accountIdentifier1}@{value1}${accountIdentifier2}@{value2}${accountIdentifier3}@{value3}.

The path uses a '$' delimiter to separate each identifier, up to a limit of three account identifiers. Each key/value is delimited by '@'.

The list of permitted account identifiers supported by the Mobile Money API can be found in the Account Identifiers section.

2.8 Account Transactions API

A disbursement organisation should use this API to return a list of transactions against their account. One of the following paths can be used:

GET /accounts/{identifierType}/{identifier}/transactions – single identifier method

or *GET /accounts/{Account Identifiers}/transactions* – multiple identifiers method

To filter the number of records returned, the following query strings can be used:

Parameter	Type	Format	Description
limit	integer	N/A	Supports pagination. If this is not supplied, then the server will apply a limit of 50 records returned for each request.
offset	integer	N/A	Supports pagination. This value will indicate the cursor position from where to retrieve the set of records. For example, a limit of 50 and offset of 10 will return records 11 to 60.

fromDateTime	string	date-time	Indicates the minimum creationDate for which records should be returned.
toDateTime	string	date-time	Indicates the maximum creationDate for which records should be returned.
transactionStatus	string	N/A	Indicates the status of the transactions to be returned.
transactionType	string	N/A	Indicates the type of the transactions to be returned.

Note 1: For a harmonised behavior, API Providers should make sure that the transactions are returned in descending date created order.

Note 2: HTTP response headers are returned with each response indicating the total number of records available (X-Records-Available-Count) and total number of records returned (X-Records-Returned-Count).

2.8.1 Account Transaction UML Class Diagram

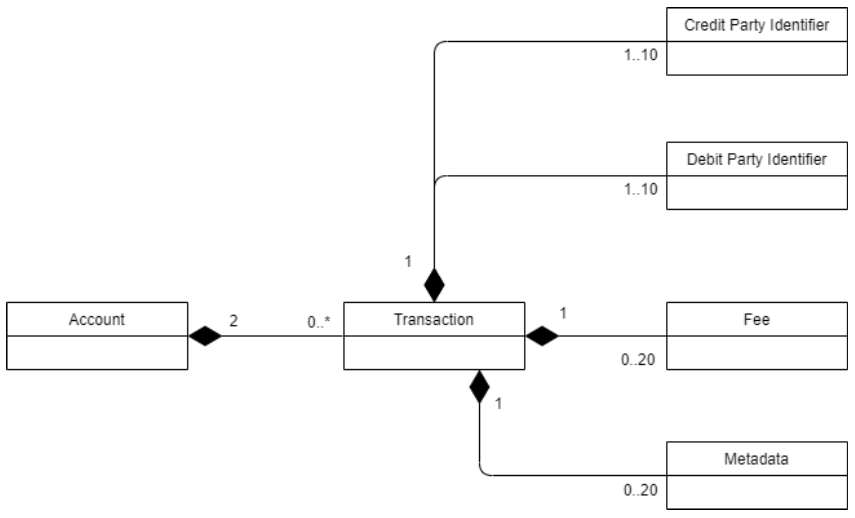

Figure 2-6 Account Transaction UML Class Diagram

2.9 Account Balances API

Using the Account Balances API, a disbursement organisation can check their balance. Permitted paths are:

GET /accounts/{identifierType}/{identifier}/balance – single identifier method

or *GET /accounts/{Account Identifiers}/balance* – multiple identifiers method

A 'self' version is also available where the calling API client is the organisation account holder. Path for the 'self' version is */accounts/balance*.

2.9.1 Account Balance UML Class Diagram

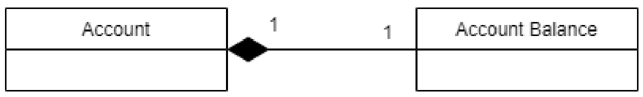

Figure 2-7 Account Balance UML Class Diagram

Balance Object					
Name	**Type**	**Description**		**Reference**	**Validation**
accountStatus	string	Indicates a harmonised representation of the account state. This will be shown as 'available' or 'unavailable'. A state of 'unavailable' means that the account is in a state that does not allow posting of transactions. Unregistered indicates that although not available, a transaction created with the account identifier(s) will result in an unregistered voucher creation.	→NA ←O		Enumeration = available, unavailable, unregistered
currentBalance	string	The current outstanding balance on the account.	→NA ←O		Please refer to API Fundamentals document for amount validation rules.
availableBalance	string	Indicates the balance that is able to be debited for an account. This balance is only provided on some API provider systems.	→NA ←O		Please refer to API Fundamentals document for amount validation rules.
reservedBalance	string	Indicates the portion of the balance that is reserved, i.e. intended to be debited. This balance is only provided on some API provider systems.	→NA ←O		Please refer to API Fundamentals document for amount validation rules.

unClearedBalance	string	Indicates the sum of uncleared funds in an account, i.e. those that are awaiting a credit confirmation.	→NA ←O		Please refer to API Fundamentals document for amount validation rules.
currency	string	Currency for all returned balances.	→NA ←O		Enumeration = ISO Currency Codes

3 Supporting Objects

3.1 Account Identifier Object

The Account Identifier object enables one or multiple identifiers to be provided to enable the recipient system to resolve the account/party.

Account Identifier Object					
Name	**Type**	**Description**		**Reference**	**Validation**
key	string	Provides the account identifier type.	→M ←M		Enumeration = Account Identifiers
value	string	Provides the account identifier type value.	→M ←M		

3.2 Metadata Object

The metadata object allows fields to be specified to convey administrative information regarding the associated resource in the form of key/value pairs. Additional fields should only be used where no suitable defined field match can be found. The number of key/value pairs is limited to 20.

Metadata Object					
Name	**Type**	**Description**		**Reference**	**Validation**
key	string	Identifies the type of additional fields.	→M ←M		
value	string	Identifies the value of the additional field.	→M ←M		

3.3 Custom Data Object

The custom data object allows additional fields to be specified for the associated resource in the form of key/value pairs. Additional fields should only be used where no suitable defined field match can be found. The number of key/value pairs is limited to 20.

Custom Data Object					
Name	**Type**	**Description**		**Reference**	**Validation**
key	string	Identifies the type of additional fields.	→M ←M		
value	string	Identifies the value of the additional field.	→M ←M		

3.4 Fees Object

An object that enables fees that are differentiated by type to be provided and/or returned.

Fees Object					
Name	**Type**	**Description**		**Reference**	**Validation**
feeType	string	Defines the type of fee.	→M ←M		
feeAmount	string	Defines the amount of the fee.	→M ←M		Please refer to API Fundamentals document for amount validation rules.
feeCurrency	string	Defines the currency for the given fee.	→M ←M		Enumeration = ISO Currency Codes

3.5 Requesting Organisation Object

An object that details the originating organisation of the request.

Requesting Organisation Object					
Name	**Type**	**Description**		**Reference**	**Validation**
requestingOrganisationIdentifierType	string	Identifies the identifier type of the requesting organisation.	→M ←M		'swiftbic', 'lei', 'organisationid'
requestingOrganisationIdentifier	string	Contains the requesting organisation identifier.	→M ←M		

4 Enumerations

4.1 ISO Currency Codes

The three-character alphabetic code for currency as defined by ISO 4217 is to be used for all currency fields. The full list of codes is maintained by Swiss Interbank Clearing on behalf of the International Organisation for Standardisation. This list can be obtained via the following website - http://www.currency-iso.org/en/home/tables/table-a1.html.

4.2 Transaction Types

A transaction type is used to classify the nature of a transaction.

Code	Description
disbursement	Disbursement of funds (making payments from an organisation (business, NGO, government entity) to a mobile money recipient.
reversal	Reversal of a prior transaction to return funds to the payer.

4.3 Account Identifiers

The Account Identifier enumeration lists all possible means to identify a target account. Identifiers can be combined if necessary, to provide a unique identifier for the target account.

Code	Short Description	Type	Description
accountcategory	Account Category	string	Can be used to identify the sources of funds category where there are multiple accounts (wallets) held against an account holder.
bankaccountno	Bank Account Number	string	Financial institution account number that is typically known by the account holder.
accountrank	Account Rank	string	Is used to identify the rank of the source of funds where there are multiple accounts (wallets) held against an account holder.
identityalias	Identity Alias	string	An alias for the identity, e.g. short code for an agent till.
iban	IBAN	string	Internationally agreed system of identifying bank accounts across national borders to facilitate the communication and processing of cross border transactions. Can contain up to 34 alphanumeric characters.

accountid	Account Holder Identity	string	Identifier for the account holder.
msisdn	MSISDN	string	Must contain between 6 and 15 consecutive digits First character can contain a '+' or digit Can contain spaces.
swiftbic	SWIFTBIC	string	A bank identifier code (BIC) is a unique identifier for a specific financial institution. A BIC is composed of a 4-character bank code, a 2-character country code, a 2-character location code and an optional 3-character branch code. BICs are used by financial institutions for letters of credit, payments and securities transactions and other business messages between banks. Please refer to ISO 9362 for further information.
sortcode	Bank Sort Code	string	Sort code to identify the financial institution holding the account.
organisationid	Organisation Account Identifier	string	Used to identify the organisation for which a payment is to be made.
username	Username	string	Used to identify target account via an associated username.
walletid	Wallet Identifier	string	A means to identify a mobile money wallet, particularly where multiple wallets can be held against an MSISDN. typically used in conjunction with MSISDN or identity alias to identify a particular wallet.
linkref	Link Reference	string	A means to uniquely identify an account via an account to account link. E.g. wallet account link to bank account.
consumerno	Consumer Number	String	Identifies the consumer associated with the account.
serviceprovider	Service Provider	String	Provides a reference for a Service Provider.
storeid	Store ID	String	Identifies the transacting store / retail outlet.
bankname	Bank Name	String	Name of the bank.
bankaccounttitle	Bank Account Title	String	The title of the bank account.

emailaddress	Email Address	String	emailaddress of the party.
mandatereference	Debit Mandate Reference	String	A means to identify an account via a debit mandate reference.

5 API Sequence Diagrams

The following sequence diagrams illustrate a selection of success and failure flows for disbursements using the Mobile Money API. For further information on API behaviour and error handling, please refer to the Mobile Money API Fundamentals document.

5.1 Individual Disbursement

This diagram illustrates an individual disbursement using an asynchronous flow with the notification provided via a callback.

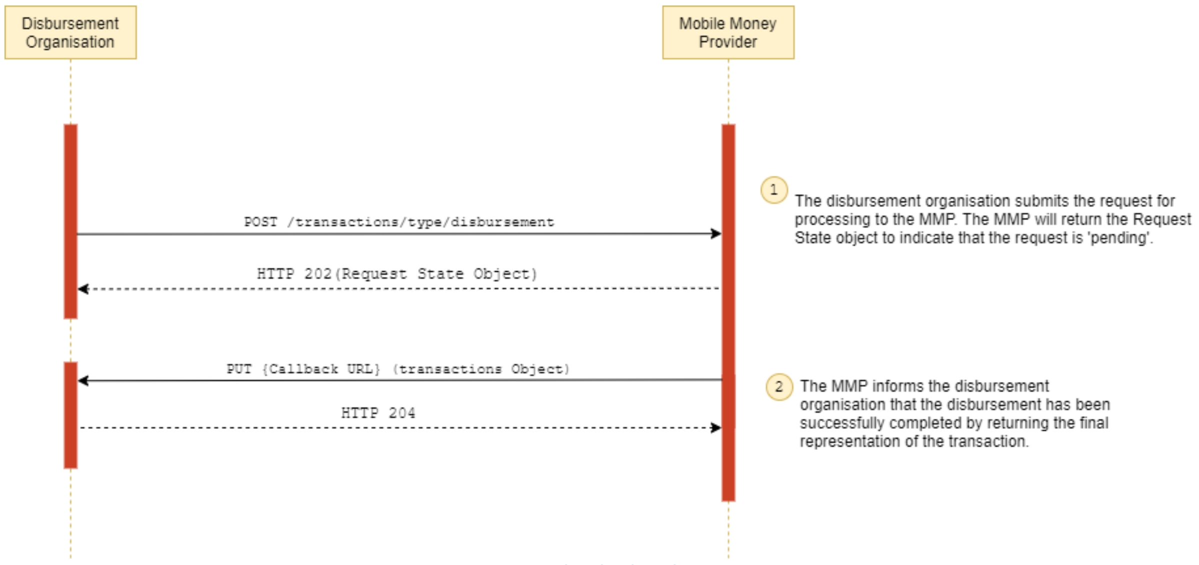
Disbursement Organisation
Mobile Money Provider
POST /transactions/type/disbursement
HTTP 202(Request State Object)
PUT {Callback URL} (transactions Object)
HTTP 204
1 The disbursement organisation submits the request for processing to the MMP. The MMP will return the Request State object to indicate that the request is 'pending'.
2 The MMP informs the disbursement organisation that the disbursement has been successfully completed by returning the final representation of the transaction.

Figure 5-1 Individual Disbursement

5.2 Individual Disbursement Failure

In this example, an asynchronous flow is used with a final callback that contains the reason for failure.

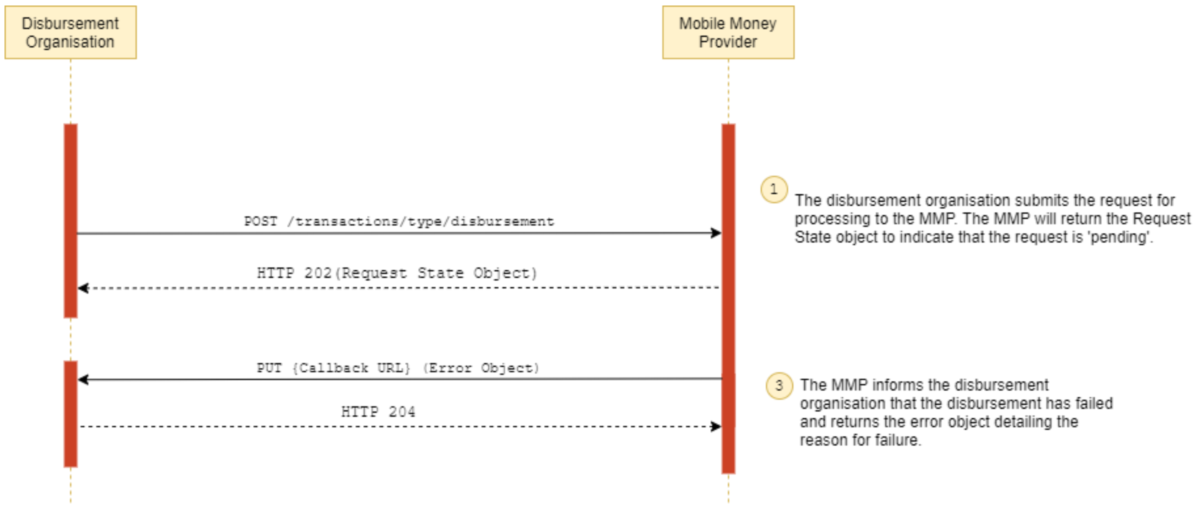
Disbursement Organisation
Mobile Money Provider
POST /transactions/type/disbursement
HTTP 202(Request State Object)
PUT {Callback URL} (Error Object)
HTTP 204
1 The disbursement organisation submits the request for processing to the MMP. The MMP will return the Request State object to indicate that the request is 'pending'.
3 The MMP informs the disbursement organisation that the disbursement has failed and returns the error object detailing the reason for failure.

Figure 5-2 Individual Disbursement Failure

5.3 Bulk Disbursement

This diagram illustrates the flow for a ‘one-shot’ bulk disbursement.

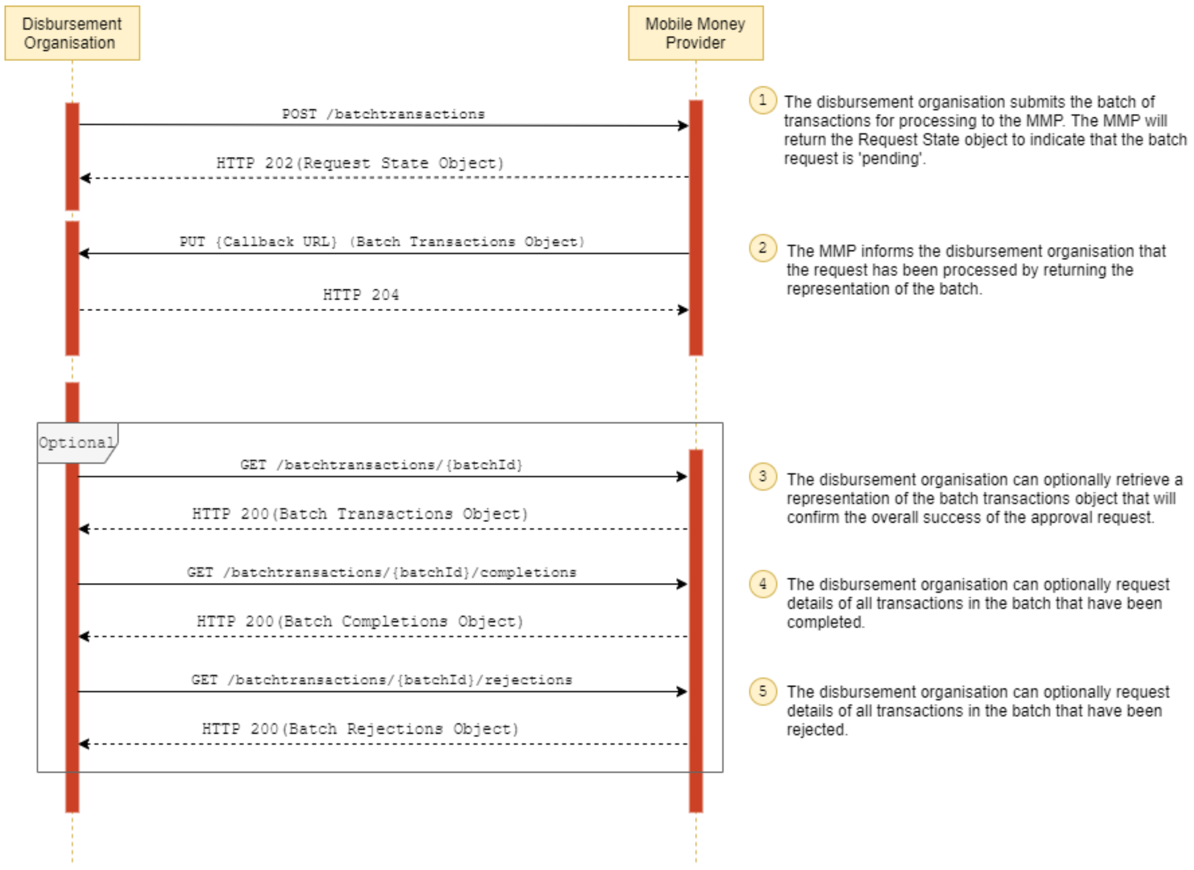
Disbursement Organisation
Mobile Money Provider
POST /batchtransactions
HTTP 202(Request State Object)
PUT {Callback URL} (Batch Transactions Object)
HTTP 204
Optional
GET /batchtransactions/{batchId}
HTTP 200(Batch Transactions Object)
GET /batchtransactions/{batchId}/completions
HTTP 200(Batch Completions Object)
GET /batchtransactions/{batchId}/rejections
HTTP 200(Batch Rejections Object)
1 The disbursement organisation submits the batch of transactions for processing to the MMP. The MMP will return the Request State object to indicate that the batch request is 'pending'.
2 The MMP informs the disbursement organisation that the request has been processed by returning the representation of the batch.
3 The disbursement organisation can optionally retrieve a representation of the batch transactions object that will confirm the overall success of the approval request.
4 The disbursement organisation can optionally request details of all transactions in the batch that have been completed.
5 The disbursement organisation can optionally request details of all transactions in the batch that have been rejected.

Figure 5-3 Bulk Disbursement

5.4 Bulk Disbursement Failure

In this example, an asynchronous flow is used with a final callback that contains the reason for failure to process the bulk request.

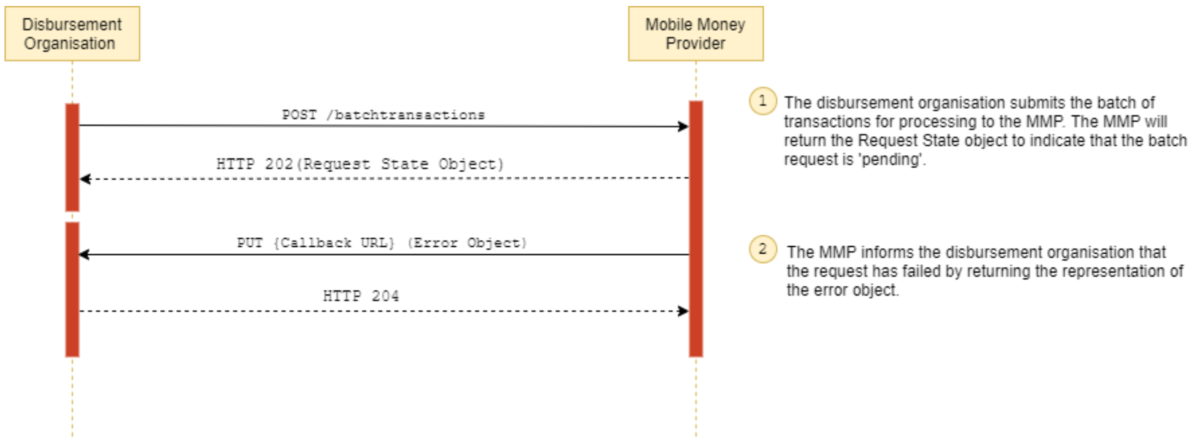
Disbursement Organisation
Mobile Money Provider
POST /batchtransactions
HTTP 202(Request State Object)
PUT {Callback URL} (Error Object)
HTTP 204
1 The disbursement organisation submits the batch of transactions for processing to the MMP. The MMP will return the Request State object to indicate that the batch request is 'pending'.
2 The MMP informs the disbursement organisation that the request has failed by returning the representation of the error object.

Figure 5-4 Bulk Disbursement Failure

5.5 Bulk Disbursement with Maker / Checker

This flow allows a bulk request to be processed in two steps. The first step involves the 'maker' system loading the request into the mobile money provider. The second step involves the 'checker' system approving the request.

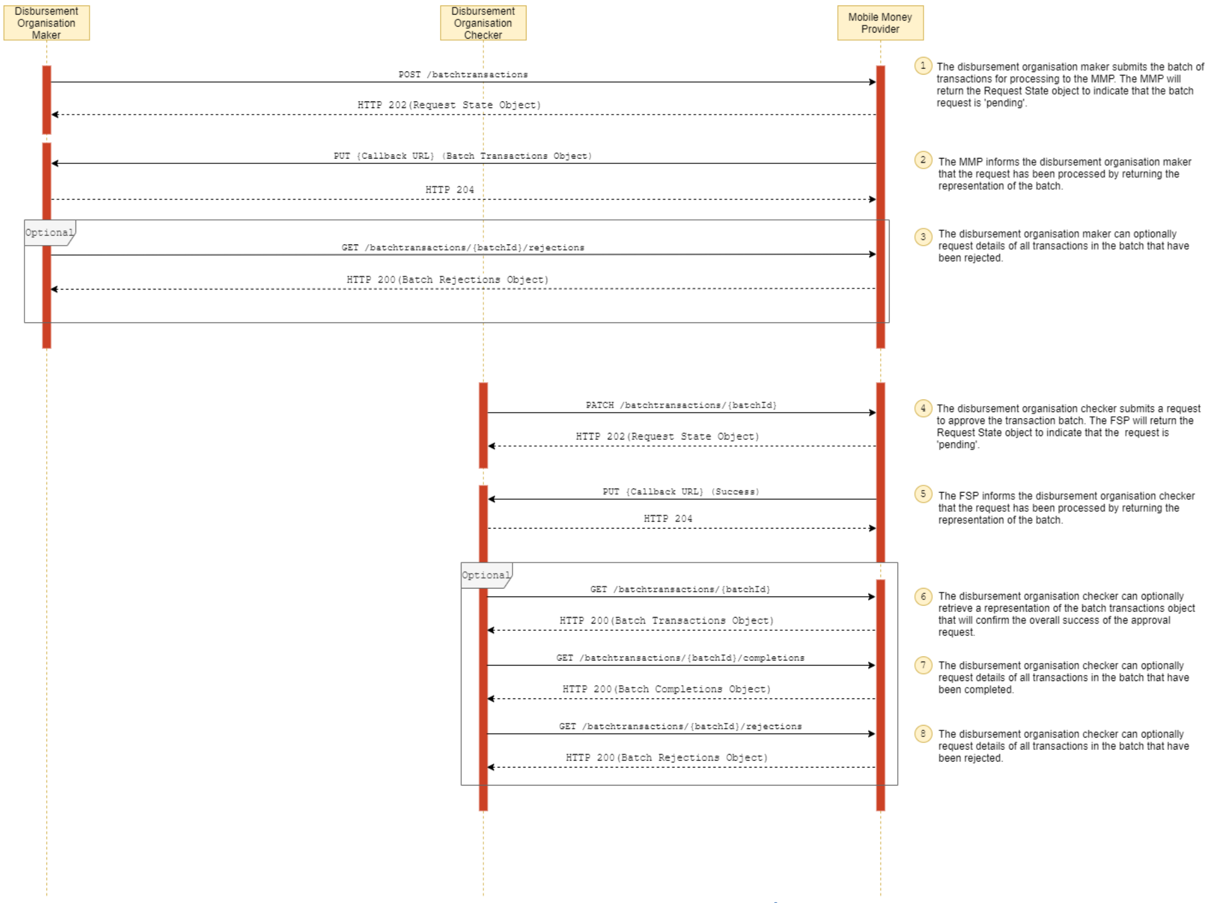
Disbursement Organisation Maker
Disbursement Organisation Checker
Mobile Money Provider
POST /batchtransactions
HTTP 202(Request State Object)
PUT {Callback URL} (Batch Transactions Object)
HTTP 204
Optional
GET /batchtransactions/{batchId}/rejections
HTTP 200(Batch Rejections Object)
PATCH /batchtransactions/{batchId}
HTTP 202(Request State Object)
PUT {Callback URL} (Success)
HTTP 204
Optional
GET /batchtransactions/{batchId}
HTTP 200(Batch Transactions Object)
GET /batchtransactions/{batchId}/completions
HTTP 200(Batch Completions Object)
GET /batchtransactions/{batchId}/rejections
HTTP 200(Batch Rejections Object)
1 The disbursement organisation maker submits the batch of transactions for processing to the MMP. The MMP will return the Request State object to indicate that the batch request is 'pending'.
2 The MMP informs the disbursement organisation maker that the request has been processed by returning the representation of the batch.
3 The disbursement organisation maker can optionally request details of all transactions in the batch that have been rejected.
4 The disbursement organisation checker submits a request to approve the transaction batch. The FSP will return the Request State object to indicate that the request is 'pending'.
5 The FSP informs the disbursement organisation checker that the request has been processed by returning the representation of the batch.
6 The disbursement organisation checker can optionally retrieve a representation of the batch transactions object that will confirm the overall success of the approval request.
7 The disbursement organisation checker can optionally request details of all transactions in the batch that have been completed.
8 The disbursement organisation checker can optionally request details of all transactions in the batch that have been rejected.

Figure 5-5 Bulk Disbursement with Maker/Checker

5.6 Individual Disbursement Using the Polling Method

In this diagram, an asynchronous flow is used with the polling method. The client polls against the request state object to determine the outcome of the individual disbursement request.

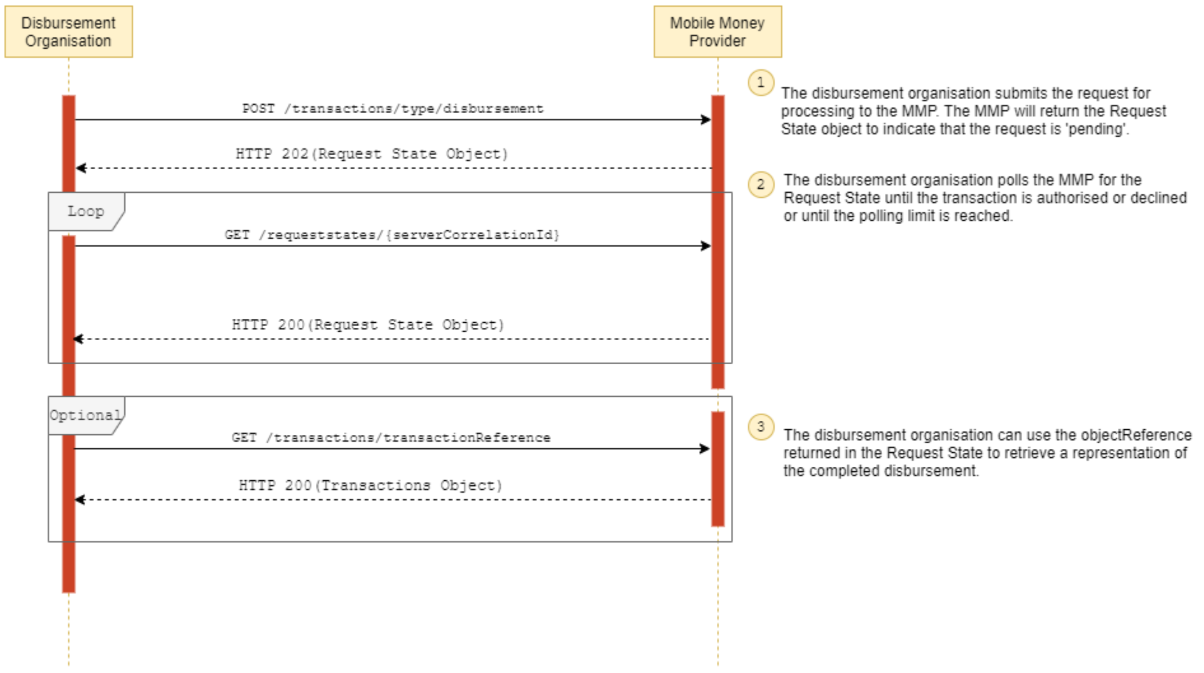
Disbursement Organisation
Mobile Money Provider
POST /transactions/type/disbursement
HTTP 202(Request State Object)
Loop
GET /requeststates/{serverCorrelationId}
HTTP 200(Request State Object)
Optional
GET /transactions/transactionReference
HTTP 200(Transactions Object)
1
The disbursement organisation submits the request for processing to the MMP. The MMP will return the Request State object to indicate that the request is 'pending'.
2
The disbursement organisation polls the MMP for the Request State until the transaction is authorised or declined or until the polling limit is reached.
3
The disbursement organisation can use the objectReference returned in the Request State to retrieve a representation of the completed disbursement.

Figure 5-6 Individual Disbursement Using the Polling Method

5.7 Disbursement Reversal

In some failure scenarios, a organisation may need to reverse an individual disbursement transaction. This diagram illustrates a reversal with the final result communicated via the callback.

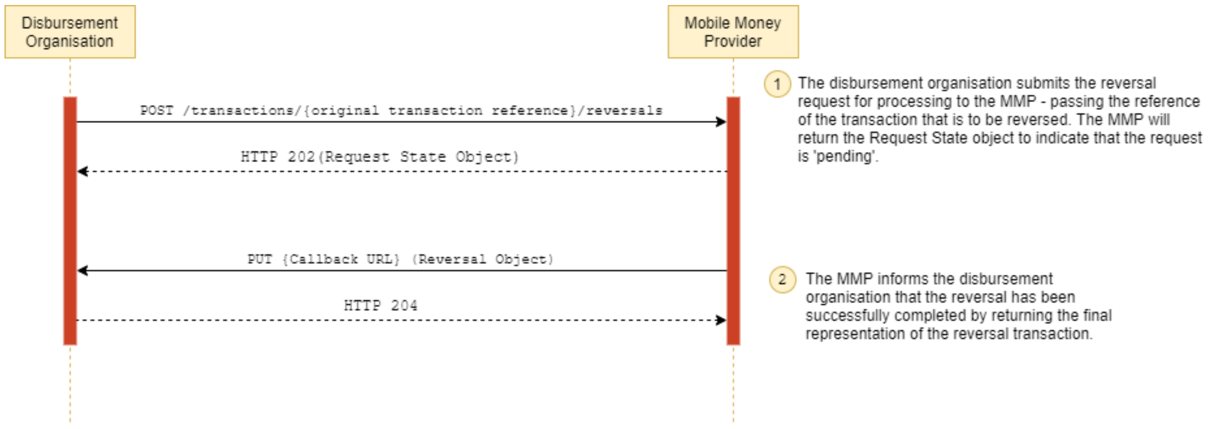
Disbursement Organisation
Mobile Money Provider
POST /transactions/{original transaction reference}/reversals
HTTP 202(Request State Object)
PUT {Callback URL} (Reversal Object)
HTTP 204
1
The disbursement organisation submits the reversal request for processing to the MMP - passing the reference of the transaction that is to be reversed. The MMP will return the Request State object to indicate that the request is 'pending'.
2
The MMP informs the disbursement organisation that the reversal has been successfully completed by returning the final representation of the reversal transaction.

Figure 5-7 Disbursement Reversal

5.8 Obtain a Disbursement Organisation Balance

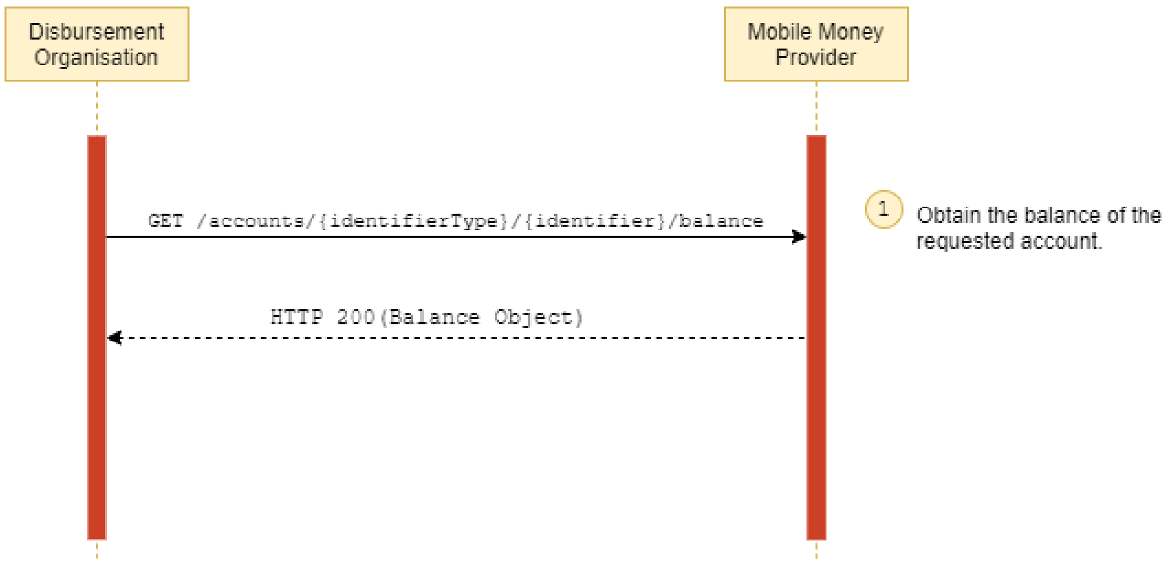
Disbursement Organisation
Mobile Money Provider
GET /accounts/{identifierType}/{identifier}/balance
HTTP 200(Balance Object)
1 Obtain the balance of the requested account.

Figure 5-8 Obtain a Disbursement Organisation Balance

5.9 Retrieve Transactions for a Disbursement Organisation

This diagram illustrates use of a cursor mechanism to retrieve all transactions for a disbursement organisation via multiple requests.

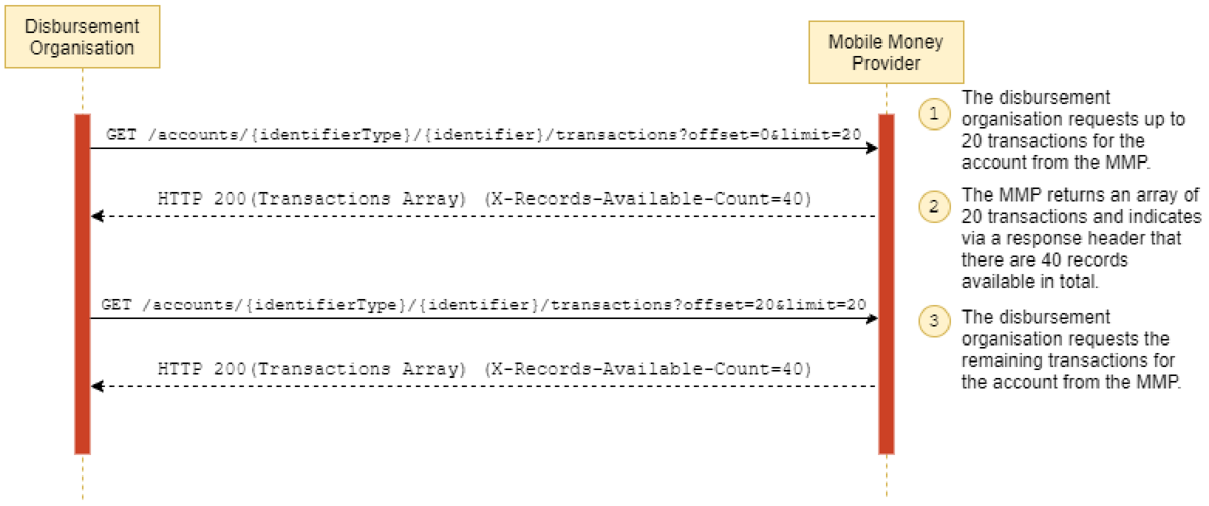
Disbursement Organisation
Mobile Money Provider
GET /accounts/{identifierType}/{identifier}/transactions?offset=0&limit=20
HTTP 200(Transactions Array) (X-Records-Available-Count=40)
GET /accounts/{identifierType}/{identifier}/transactions?offset=20&limit=20
HTTP 200(Transactions Array) (X-Records-Available-Count=40)
1 The disbursement organisation requests up to 20 transactions for the account from the MMP.
2 The MMP returns an array of 20 transactions and indicates via a response header that there are 40 records available in total.
3 The disbursement organisation requests the remaining transactions for the account from the MMP.

Figure 5-9 Retrieve Transactions for a Disbursement Organisation

5.10 Check for Service Availability

The Heartbeat API is used for monitoring purposes and establishes whether the mobile money provider is in a state that enables a client to submit a request for processing.

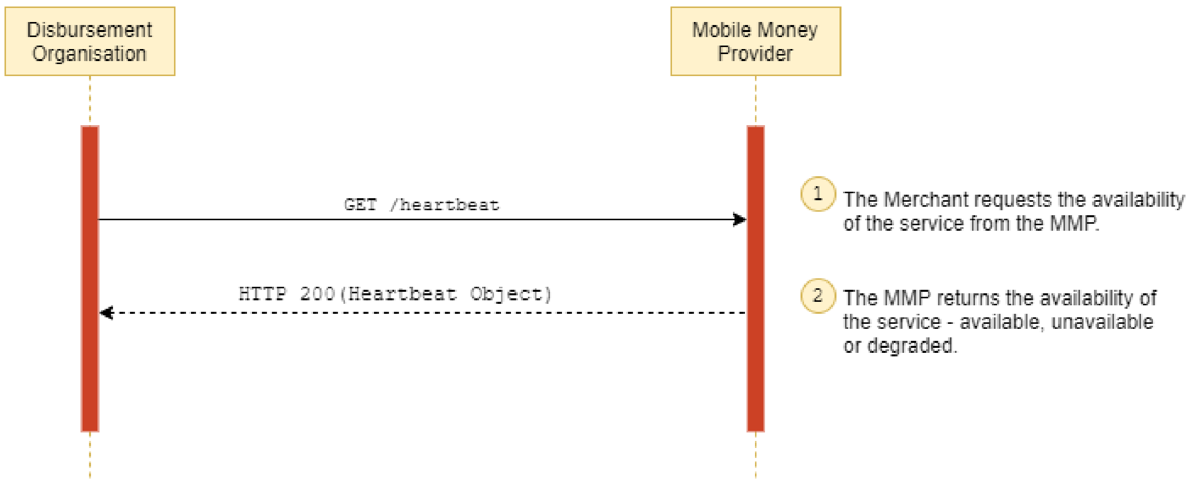
Disbursement Organisation
Mobile Money Provider
GET /heartbeat
HTTP 200(Heartbeat Object)
1 The Merchant requests the availability of the service from the MMP.
2 The MMP returns the availability of the service - available, unavailable or degraded.

Figure 5-10 Check for Service Availability

5.11 Retrieve a Missing API Response

This API can be used by the disbursement organisation to retrieve a link to the final representation of the resource for which it attempted to create. Use this API when a callback is not received from the mobile money provider.

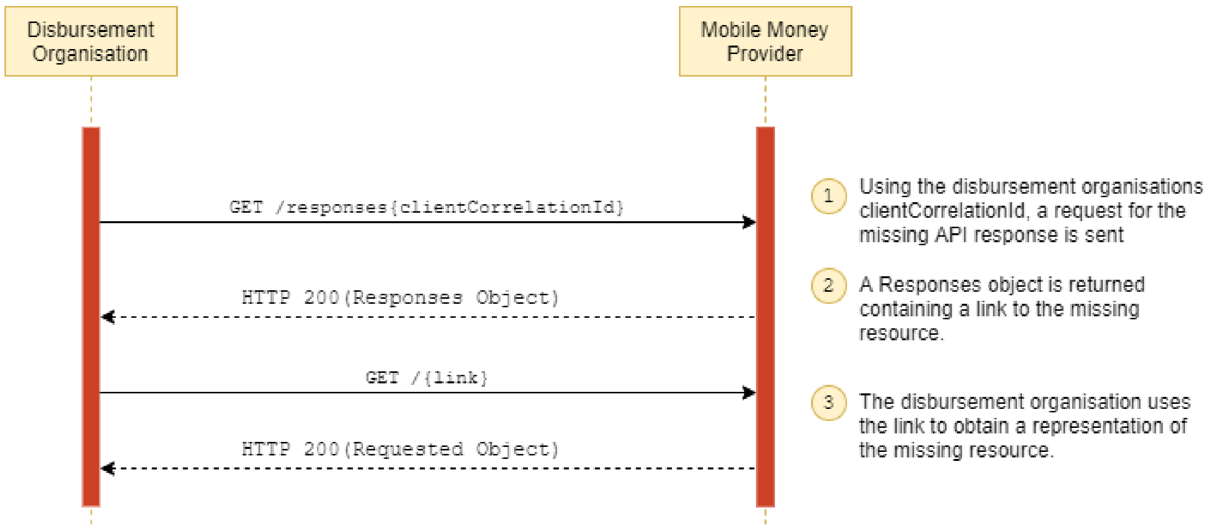
Disbursement Organisation
Mobile Money Provider
GET /responses{clientCorrelationId}
HTTP 200(Responses Object)
GET /{link}
HTTP 200(Requested Object)
1 Using the disbursement organisations clientCorrelationId, a request for the missing API response is sent
2 A Responses object is returned containing a link to the missing resource.
3 The disbursement organisation uses the link to obtain a representation of the missing resource.

Figure 5-11 Retrieve a Missing API Response

This document is produced by the GSMA with input from the GSMA Mobile Money API Working Group. It is our intention to provide a quality product for your use. If you find any errors or omissions, please contact us with your comments. You may notify us at support.mmapi@gsma.com.