Mobile Money API Specification 1.2.0 Merchant Payments

Document Summary

Official Document Number, Document Title and Version Number	Mobile Money API Specification 1.2.0 – Merchant Payments
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 2020-7, RFC 2020-10, RFC 2020-13, RFC 2020-17, RFC 2020-18	GSMA

Other Information

Type	Description
Document Owner	Mobile Money API Merchant Payments 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 Merchant Payment Mobile Money APIs, which are a subset of the GSMA Mobile Money API Specification. The Merchant Payment Mobile Money APIs allow merchants to accept payments from mobile money customers. Supported payment mechanisms include:

- **Payee-initiated merchant payment**. The merchant initiates the payment and the payer is requested to authenticate to confirm acceptance by the mobile money provider.
- **Payer-initiated merchant payment**. The payer initiates the payment by specifying the merchant that is to be paid.
- **Merchant payment via pre-authorised payment code**. The payer generates a payment authorisation code up to a maximum payment amount. The merchant then enters or scans (if rendered as a QR code) the payer's code to perform the payment.

Closed loop and open-loop merchant payments are supported by the Mobile Money API. Closed loop merchant payments occur where the payer and payee accounts reside with the same mobile money provider. Open loop payments occur where the payer and payee accounts reside with different mobile money providers.

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:

1.1 Intended Audience

Audience	Usage	Role
Mobile Money Providers	• To understand how to implement the Mobile Money API to receive payments from merchants.	API Provider
Merchants	• To understand how to implement the Mobile Money API to accept mobile money payments.	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

Merchant payments can be created and viewed using Transactions APIs.

The following paths are permitted:

Operation	Path	Description
Create	*POST /transactions*	For creation of a transaction where transaction *type* is specified in the body.
	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.

2.1.1 Transaction UML Class Diagram

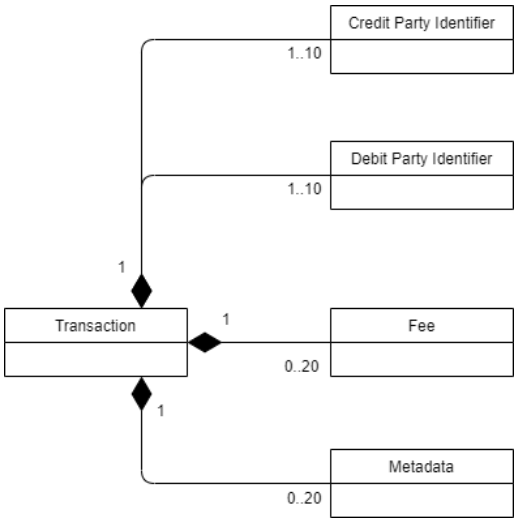

Figure 2-1 Transaction UML Class Diagram

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	**/transactions API:** Mandatory **/transactions/type API:** 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	**/transactions API:** Mandatory **/transactions/type API:** 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	→O ←O		

		according to Provider.			
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 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 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 Object	
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	→NA ←O		

		differ from the Transaction Reference.			
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 merchant payment. The originating transaction reference must be provided in the path in order to identify the payment to be reversed. For a partial reversal, the amount needs to be supplied.

For viewing or updating reversals, the Transactions API should be used. For performing a refund without the original transaction reference, use the Transactions API.

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

2.2.1 Reversal UML Class Diagram

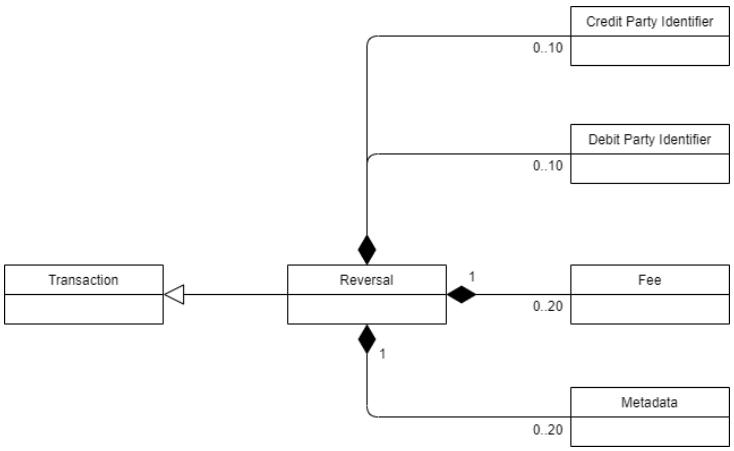

Figure 2-2 Reversal UML Class Diagram

2.2.2 Reversal Object Definition

Reversal Object					
Name	**Type**	**Description**		**Reference**	**Validation**
transaction Reference	string	Unique reference for the transaction. This is returned in the response by API provider.	→NA ←M		
requesting OrganisationTransactionReference	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 Object	
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 Merchant Account APIs

Using the mobile money APIs, merchants can:

- View payments for their account.
- View their account balance.

2.3.1 Identifying a Merchant Account

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

2.3.1.1 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}*.

2.3.1.2 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.

2.4 Account Transactions API

A merchant should use this API to return a list of payments 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. This can be 'merchantpay', 'reversal' or 'adjustment'

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).

2.4.1 Account Transaction UML Class Diagram

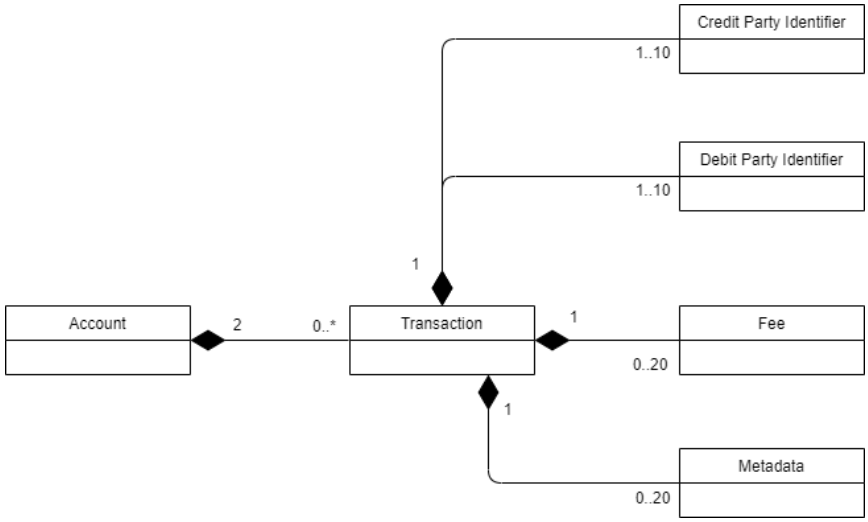

Figure 2-3 Account Transaction UML Class Diagram

2.5 Account Balances API

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

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

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

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

2.5.1 Account Balance UML Class Diagram

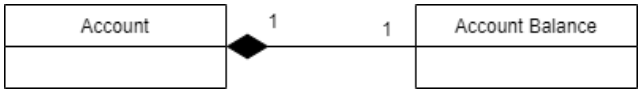

Figure 2-4 Account Balance UML Class Diagram

2.5.2 Account Balance Object Definition

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.
available Balance	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	→NA ←O		Please refer to API Fundamentals document for

		balance is only provided on some API provider systems.			amount validation rules.
unCleared Balance	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

2.6 Authorisation Codes API

The Authorisation Codes APIs allow a payer to generate a payment code which when presented to the payee, can be redeemed for an amount associated with the code.

Authorisation codes can be set to expire. Note that expiry time can be specified via the API, however the mobile money provider may mandate a common expiry period for all codes.

The following paths are permitted:

- **Generate** an Authorisation Code. *POST /accounts/{identifierType}/{identifier}/authorisationcodes* or *POST /accounts/{Account Identifiers}/authorisationcodes*
- **Cancel** an Authorisation Code*. PATCH /accounts/{identifierType}/{identifier}/authorisationcodes/{authorisationCode}* or *PATCH /accounts/{Account Identifiers}/authorisationcodes/{authorisationCode}*.
- **View** Authorisations Codes for a given account. *GET /accounts/{identifierType}/{identifier}/authorisationcodes/{authorisationCode}* or *GET /accounts/{Requestor Account Identifiers}/authorisationcodes/{authorisationCode}*.
- **View** all Authorisation Codes for a given account. *GET /accounts/{identifierType}/{identifier}/authorisationcodes* or *GET /accounts/{Requestor Account Identifiers}/authorisationcodes*.

When retrieving authorisation codes, the following query string parameters can be used to filter the number of records returned:

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.
codeState	string	string	Allows filtering on the state of the authorisation code.

Note 1: For a harmonised behavior, API Providers should make sure that authorisation codes 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).

2.6.1 Authorisation Code UML Class Diagram

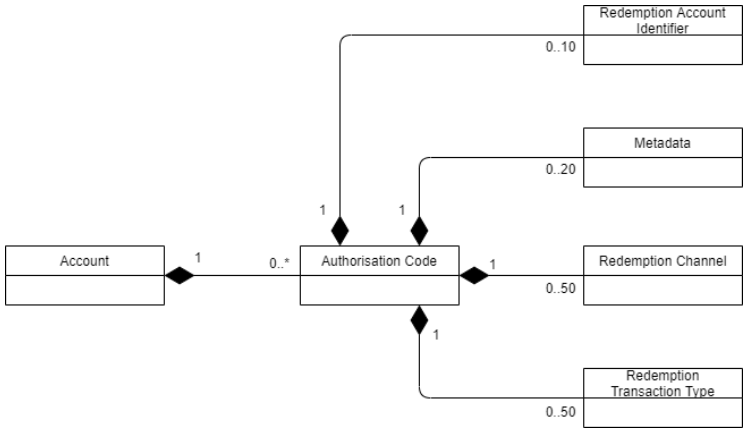

Figure 2-5 Authorisation Code UML Class Diagram

2.6.2 Authorisation Code Object Definition

Authorisation Codes Object					
Name	**Type**	**Description**		**Reference**	**Validation**
authorisation Code	string	The code that will be presented to the other party for redemption.	→NA ←M		
codeState	string	Indicates the state of the Authorisation Code.	→NA ←M		Enumeration = 'active', 'expired', 'cancelled'
amount	string	Indicates the amount associated with the authorisation code. Typically, this is set by the client.	→O ←O		Please refer to API Fundamentals document for amount validation rules.
currency	string	Indicates the amount currency. Must be supplied when an amount is supplied.	→O ←O		Enumeration = ISO Currency Codes
amountType	string	The amount for the authorisation can be an exact amount or can be a maximum amount, i.e. redemption up to but not higher than the amount specified.	→O ←O		Enumeration = 'exact', 'maximum'

codeLifetime	integer	Indicates the expiry time in seconds of the code. Depending upon the use case, this can be set by the client or server.	→O ←O		If supplied, then must be 1 second or greater.
holdFundsIndicator	boolean	Indicates whether funds should be reserved against the payer's account where the payer is the requestor.	→O ←O		
redemptionAccountIdentifiers	array	A series of key/value pairs that identify the account where the code must be redeemed. Only needed if the redemption account needs to be explicitly stated.	→O ←O	Account Identifiers	
redemptionChannels	string	Indicates the channel(s) that the code can be redeemed against, e.g. ATM, Merchant, etc..	→O ←O	Channel Types Object	
redemptionTransactionTypes	string	Indicates the Transaction Types(s) that the code can be redeemed against.	→O ←O	Transaction Types Object	
requestingOrganisation	object	The originating organisation of the request.	→O ←O	Requesting Organisation Object	
creationDate	date-time	Indicates when the link was created as recorded by the API provider.	→NA ←O		
modificationDate	date-time	Indicates when the link was modified as recorded by the API provider.	→NA ←O		
requestDate	date-time	The date and time of the request as provided 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	→O ←O	Metadata	

		information regarding the authorisation code.			

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		

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 Transaction Type Object

This object enables multiple transaction types to be specified along with paired sub-types. This object is used where multiple transaction types need to be passed in an API.

Transaction Type Object					
Name	**Type**	**Description**		**Reference**	**Validation**
transactionType	string	Identifies the Transaction Type.	→M ←M		Enumeration = Transaction Types
transactionSubType	string	Identifies the Transaction Sub-Type.	→O ←O		

3.5 Channel Type Object

This object enables multiple channel types to be specified. This object is used where multiple channel types need to be passed in an API.

Channel Type Object					
Name	**Type**	**Description**		**Reference**	**Validation**
channelType	string	Identifies the Channel Type.	→M ←M		

3.6 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

3.7 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
merchantpay	Purchases of goods and/or services from shops (payer present) or online (payer not present).
adjustment	General adjustments to an account via an adjustment transaction (e.g. refunds).
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 merchant payments 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 Payee-Initiated Merchant Payment

In this example, an asynchronous payment flow is used with a final callback. The merchant initiates the request and will be credited when the payer approves the request.

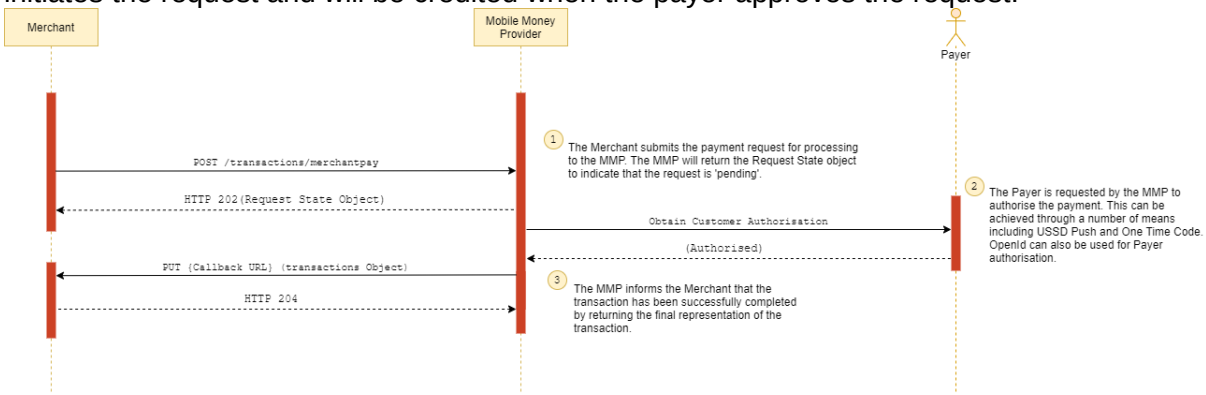

Figure 5-1 Payee Initiated Merchant Payment

5.2 Payee-Initiated Merchant Payment Failure

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

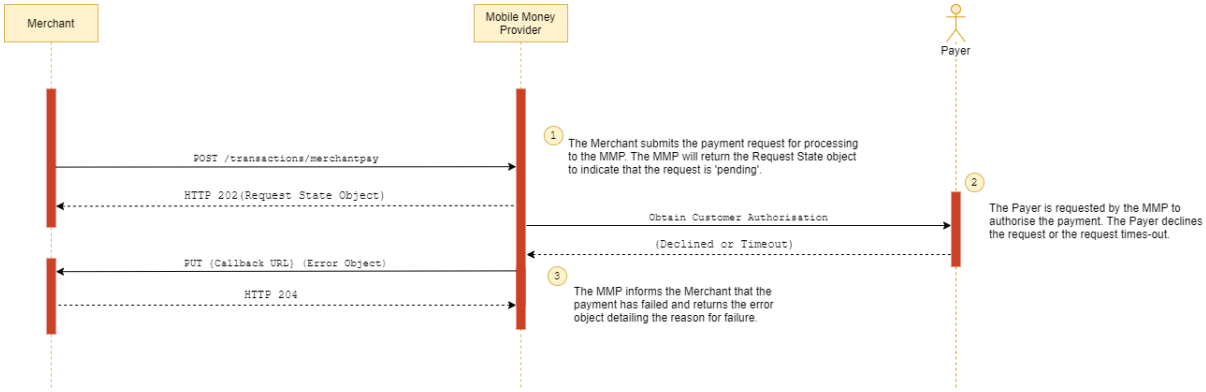

Figure 5-2 Payee Initiated Merchant Payment Failure

5.3 Payee-Initiated Merchant Payment using the Polling Method

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

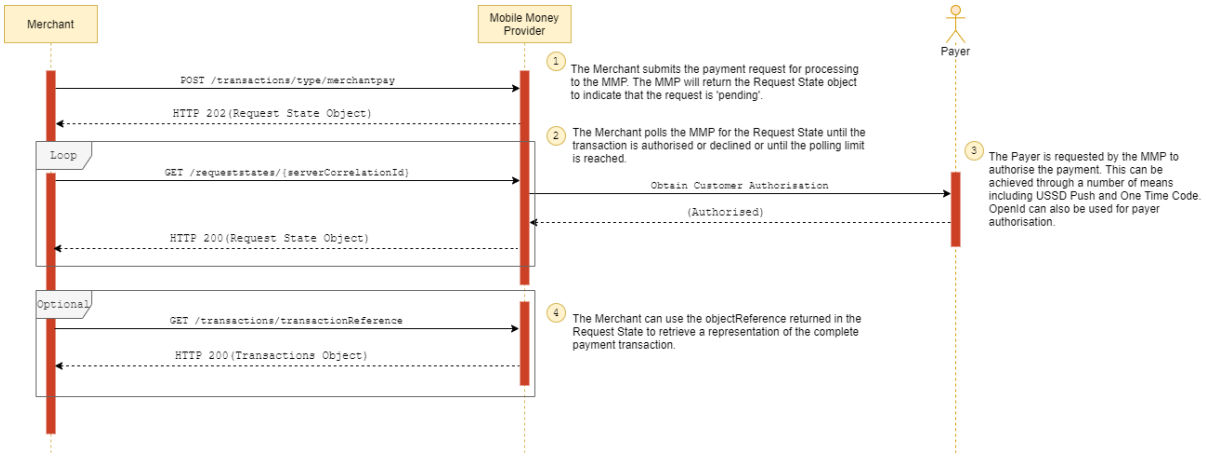

Figure 5-3 Payee Initiated Merchant Payment Using the Polling Method

5.4 Payer-Initiated Merchant Payment

In this example, an asynchronous payment flow is used with a final callback. The payer initiates the request and will be debited upon successful completion of the request.

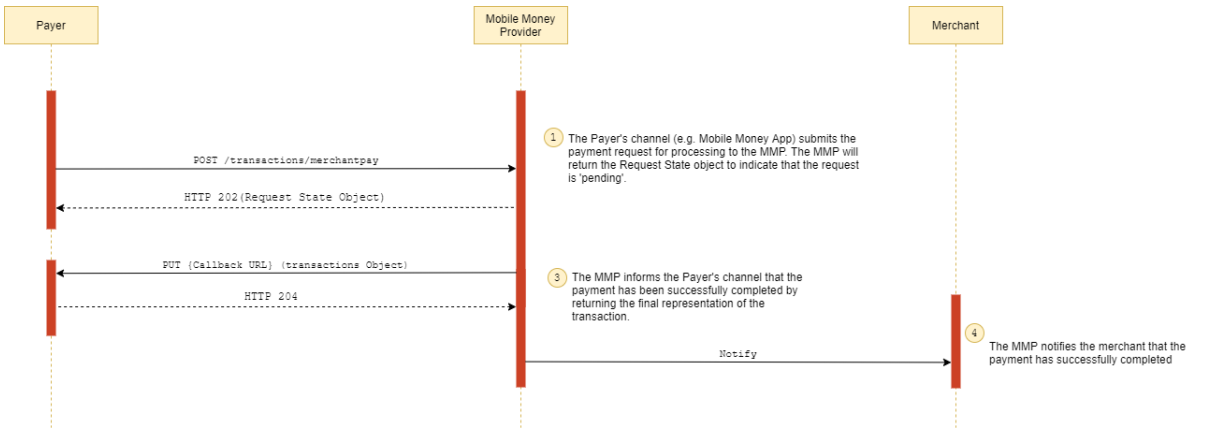

Figure 5-4 Payer Initiated Merchant Payment

5.5 Payer-Initiated Merchant Payment Failure

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

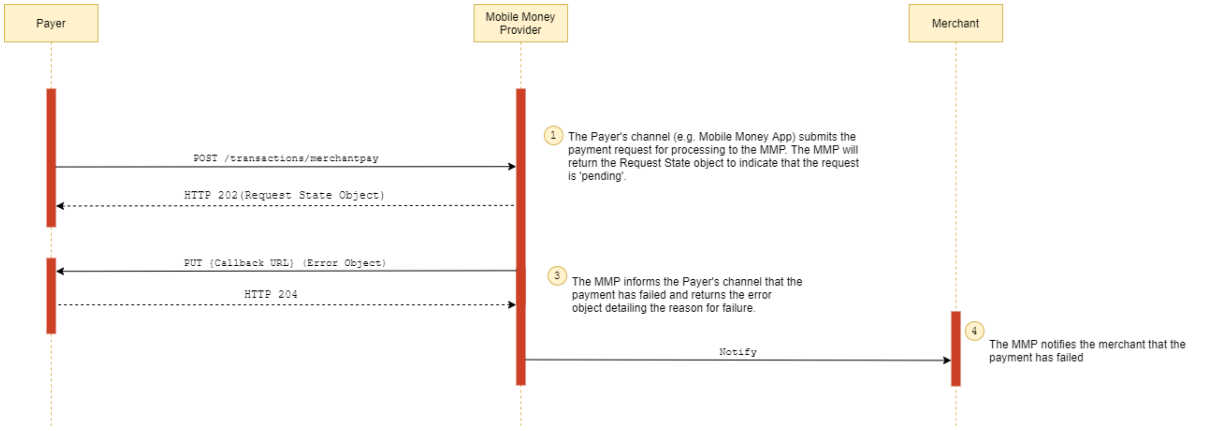

Figure 5-5 Payer Initiated Merchant Payment Failure

5.6 Payee-Initiated Merchant Payment using a Pre-authorised Payment Code

In this example the /authorisationcodes API is used to obtain a pre-authorised payment code. This in turn is presented by the payer to the merchant who initiates the payment request. Both flows in the diagram result in a callback. This flow is primarily used for payment on delivery use cases.

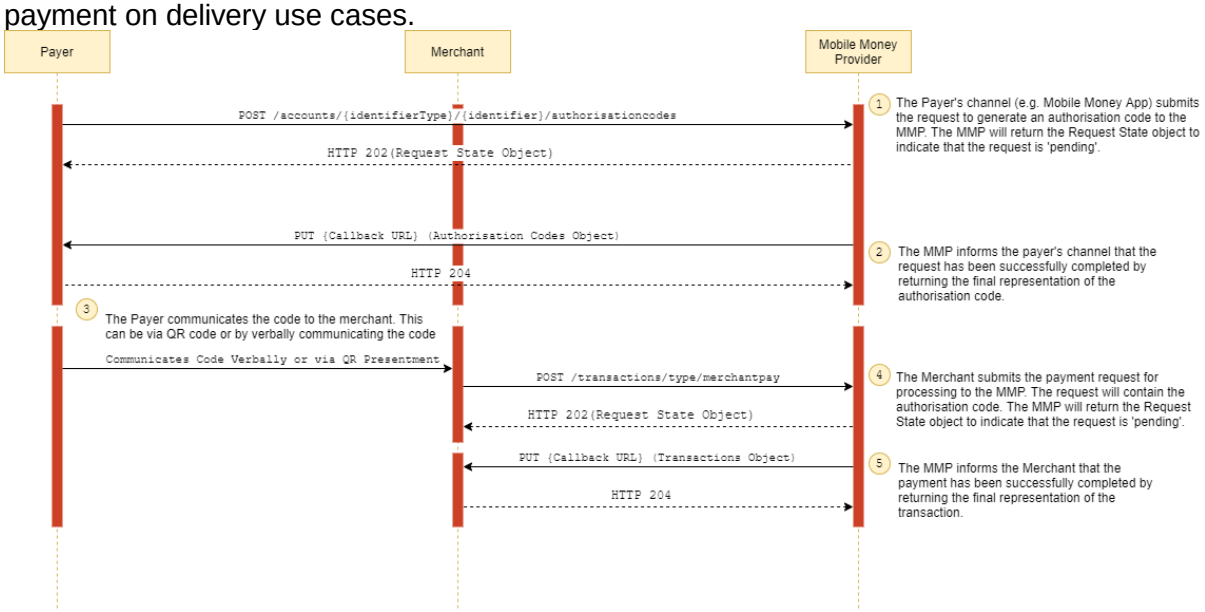

Figure 5-6 Payee Initiated Merchant Payment using a Pre-authorised Payment Code

5.7 Merchant Payment Refund

Merchants can issue a refund to payers. In this diagram, the refund is not linked to the original transaction and hence the /transactions API is used. Where a refund needs to be linked to the original transaction, the /reversals API must be used to perform the refund.

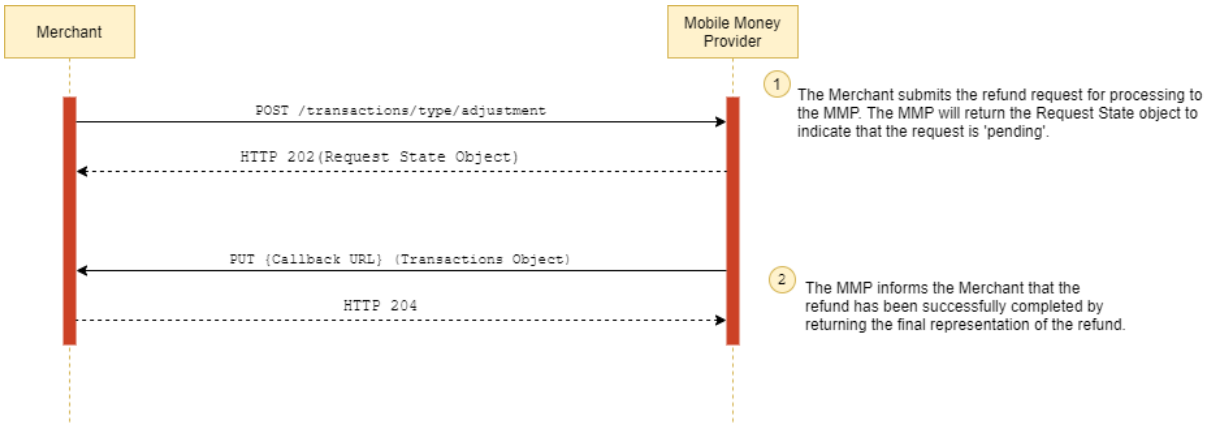

Figure 5-7 Merchant Payment Refund

5.8 Merchant Payment Reversal

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

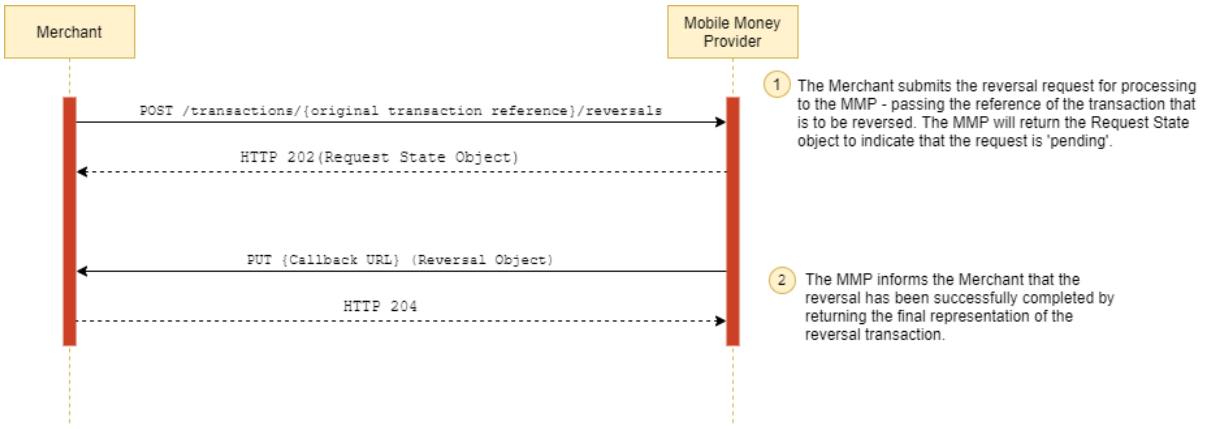

Figure 5-8 Merchant Payment Reversal

5.9 Obtain a Merchant Balance

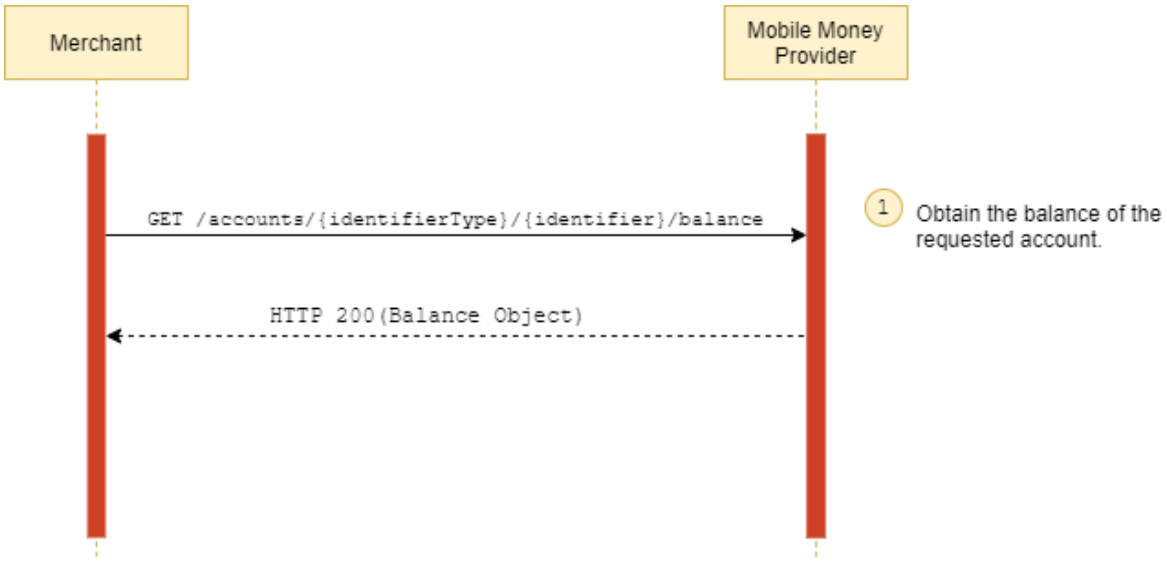

Figure 5-9 Obtain a Merchant Balance

5.10 Retrieve Payments for a Merchant

This diagram illustrates use of a cursor mechanism to retrieve all payments for a merchant via multiple requests.

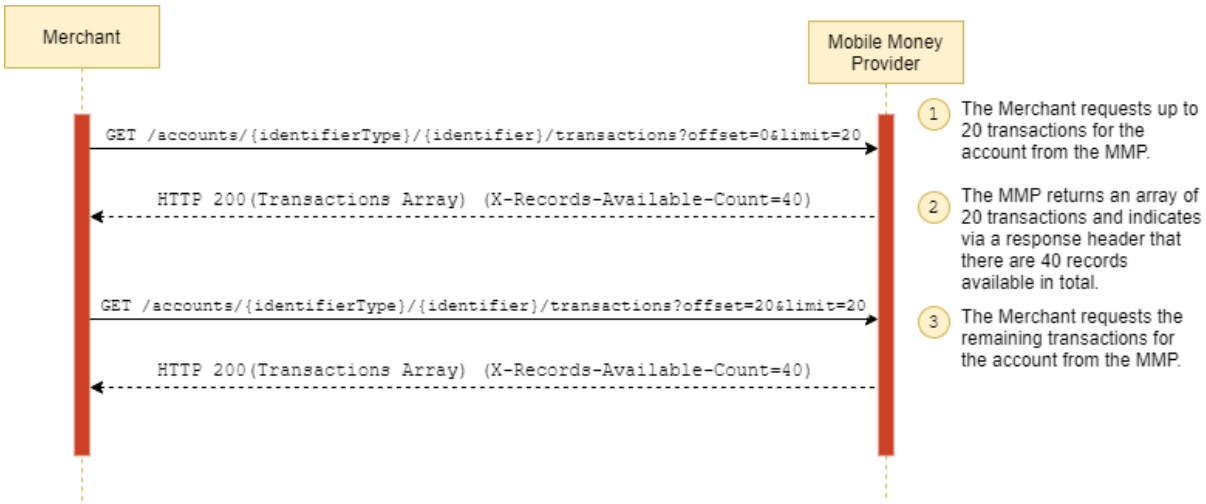

Figure 5-10 Retrieve Payments for a Merchant

5.11 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.

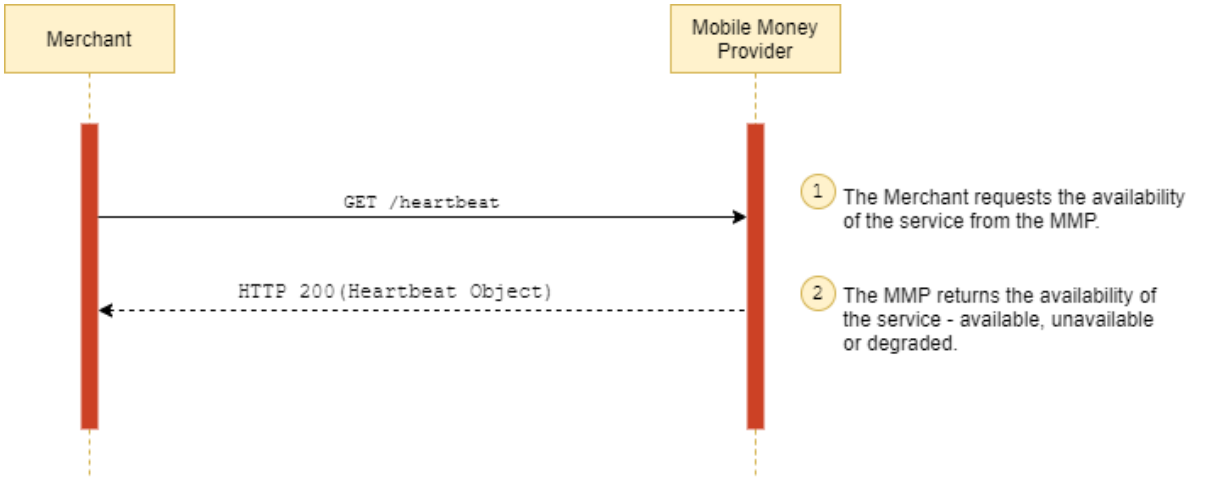

Figure 5-11 Check for Service Availability

5.12 Retrieve a Missing API Response

This API can be used by the merchant 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.

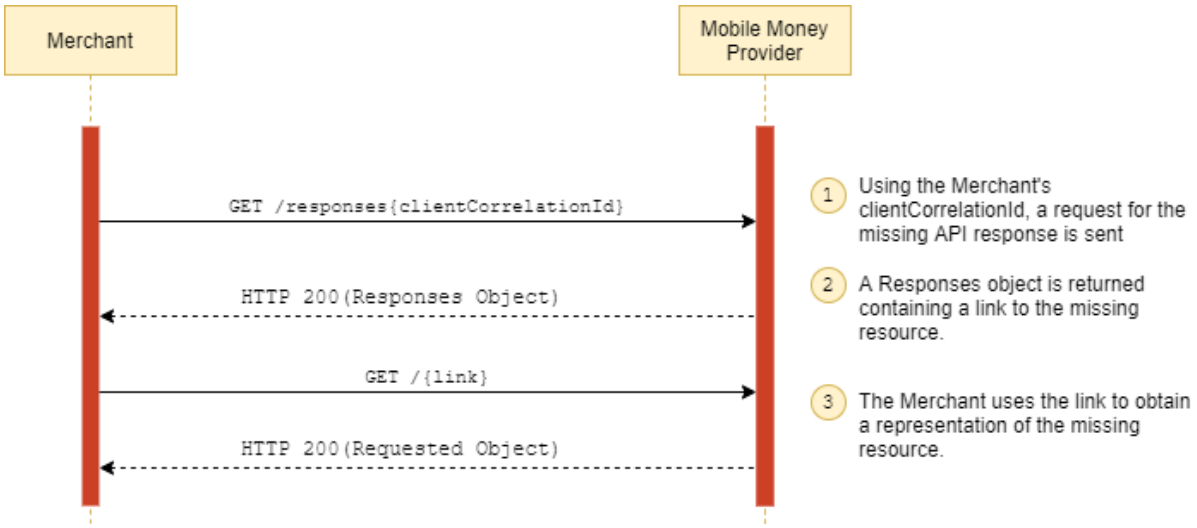

Figure 5-12 Retrieve a Missing API Response

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.