Change Request Form	
	**Mobile Money API 1.1.2 Specification Description**

Document Summary	
Official Document Number, Document Title and Version Number	Mobile Money API 1.1.2 Specification Description
Official Document Type	Non-binding Permanent Reference Document
Change Request Security Classification	Non-confidential
Is this a new document or a Major or Minor Change?	Minor Update
Will this Change Request result in a Major or Minor version update?	Minor Version
This document is for	Discussion
Input Editor and Organisation	
Additional Contributors	[List of contributors]
Issuing Group/Project	[Issuing Group/Project]
Approving Group/Project	[Approving Group/Project]
Change Request Creation Date	31/03/2020
What are the reasons for and benefits of creating this new document or Change Request?	Accessible APIs are a core requirement to ecosystem growth. However, currently, the API landscape is scattered, inconsistent and fragmented making it hard for operators and 3rd parties to interconnect efficiently – an issue a harmonised API can solve.

Review Log (to be completed by GSMA Support Staff)			
Workflow Step	***Document Review Comments***	***GSMA Support Staff Name***	***Comments Date***
Step 1: Change Request Creation (no comments required)			
Step 2: Document Quality and/or Legal Review			
Document Quality Team	INSERT COMMENTS HERE Please enter details for the Quality Review Confirm Document Quality Team feedback Record any issues, actions and key decisions	GSMA Support Staff Name	DD/MM/YY
Legal Review	INSERT COMMENTS HERE Please enter details for the Legal Review Confirm Legal feedback Record any issues, actions and key decisions	GSMA Support Staff Name	DD/MM/YY
Step 3: Formal Review			
Group(s)/Project(s) Review(s) Comments and Feedback	INSERT COMMENTS HERE Please enter details for the Group(s)/Project(s) Review(s) Record any issues, actions and key decisions Confirm outcome of Formal Review	GSMA Support Staff Name	DD/MM/YY
Step 4: Formal Approval(s)			
Group(s)/Project(s) Approval(s) Comments and Feedback	INSERT COMMENTS HERE Please enter details for the Group(s)/Project(s) Approval(s) Record any issues, actions and key decisions Confirm outcome of Formal Approval	GSMA Support Staff Name	DD/MM/YY

Document History

Version	Date	Brief Description of Change	Approval Authority	Editor / Company
1.0	Aug 2016	• 1.0 Specification Description	GSMA	GSMA
1.1.0 Beta	2018 – Dec 2019	• Aligned property names in this document to the actual properties as held in the Swagger definition to remove confusion • Changed correlation ID mandatory to optional. • Added X-User-Credential-1 and X-User-Credential-2. Only for use when Open ID is not possible for end-user authentication. • Added X-Channel which can be used to identify the API invocation channel.	GSMA, Mobile Money API Industry	GSMA

		• Generally aligned headers with the security specification. • Clarified difference between request and response custom headers • Changed all instances of '*Date Created*' to '*Creation Date*' and '*Date Modified*' to '*Modification Date*' in the document to align with the Swagger definition. • Added Bill Status property to Bills API • Added Payment Type property to Bill Payments API • Added Bill Companies APIs • Added new Account Identifiers - consumerno, serviceprovider, storeid, bankname, bankaccounttitle, emailaddress, mandateReference • Added AuthorisationCodes APIs • Add Transaction Fees to Transactions, Reversals and Quotations APIs • Added new transaction filters – Type and Status • Modified the MSISDN regular expression (refer to Swagger) • Added clarity on valid formats relating to regular expressions • Added new API to enable the Transaction Type to be passed in the URL for a POST. Changed Callback Method: • Added X-Callback-URL to the request header. • Replace PATCH callback with PUT. • Returned actual representation of the created resource in the callback. Updated Bills/Payments Properties: • Added 7 new properties: serviceProviderPaymentReference billPaymentStatus requestingOrganisationTransactionReference requestingOrganisation metadata serviceProviderComment serviceProviderNotification • Renamed amountPaid to paidAmount		

		• Set maximum number of transactions that can be supplied in Batch Transactions API. • Changed processingFlag on Batch Transactions API from optional to not applicable in the request payload. • Changed mandateReference on Debit Mandates API from optional to not applicable in the request payload. • Links API – changed mode from optional to mandatory. • Changed senderBlockingReason and RecipientBlockingReason from optional to not applicable in the Quotations API request payload. • Increased decimal places for fxRate from 2 to 10. • Replaced /msisdn/{msisdn} path construct with a more flexible construct that caters for all account identifier types: /{identifierType}/{identifier} • Added new 'self' API for Account Balance. • Change requestdate on transactions and reversals APIs to be optional on request and response. • Changed X-Date header to be optional. • Add the ability for clients to discretionally provide credit and/or debit party where the calling client is one of the parties: ○ Debit and Credit party can be supplied where the calling client is neither credit nor debit party ○ Debit Party only can be supplied where the calling client is the credit party ○ Credit Party only can be supplied where the calling client is the debit party		
1.1.0	Feb 2020	• Changed amount validation as per RFC-1. • Implemented headers - X-Account-Holding-Institution-Identifier and X-Account-Holding-Institution-Identifier-Type as per RFC-2 • General improvements to spec document, corrections.	GSMA Mobile Money API Technical Steering Committee	GSMA

		• Formatting, template updates.		
1.1.2	March 2020	• Added missing Security Customer Header – X-User-Bearer • Added explanatory notes to X-API-Key custom header	GSMA	GSMA
1.1.3	April 2020	• Fixed a small number of property typos (corrected uppercase on first character)		

Other Information

Type	Description
Document Owner	GSMA Mobile Money API Technical Steering Committee
Editor / Company	GSMA

Table of Contents

1 Introduction 8
1.1 Definitions of Terms 8
2 API Quick Start Guide 11
3 API Fundamentals 12
3.1 URI 12
3.2 Methods 12
3.3 Patch Specifics 12
3.4 Resource Naming 12
3.5 Identifying the Resource 13
3.6 Client Correlation ID 13
3.7 Use Case Flow Patterns 14
3.8 Case Sensitivity 14
3.9 HTTP Header Information 14
3.9.1 Standard Request Headers 14
3.9.2 Standard Response Headers 15
3.9.3 Custom Request Headers 15
3.9.4 Custom Response Headers 16
3.10 API Versioning 17
3.11 Amount Validation 18
4 API Service Definition 19
4.1 Transactions API 21
4.2 Reversals API 24
4.3 Introduction to Processing Batch Transactions 27
4.3.1 One-Shot Batch Processing 27
4.3.2 Batch Processing with Maker/Checker 28
4.4 Batch Transactions API 28
4.5 Batch Rejections API 30
4.6 Batch Completions API 31
4.7 Introduction to Accounts APIs 33
4.7.1 Identifying a Target Account 33
4.7.2 Supported Account Resources 33
4.8 Account Transactions API 34
4.9 Accounts Status API 34
4.10 Account Balances API 35
4.11 Account Holder Name API 36
4.12 Statement Entries API 36
4.13 Bills API 38
4.14 Bills Payments API 39
4.15 Bill Companies API 40
4.16 Debit Mandates API 41
4.17 Links API 43
4.18 Quotations API 44

4.19 Authorisation Codes API 47
5 Supporting Objects 49
5.1 International Transfer Information Object 49
5.2 KYC Information Object 50
5.3 Name Object 51
5.4 ID Document Object 51
5.5 Address Object 52
5.6 Account Identifiers Object 52
5.7 Quotes Object 53
5.8 Metadata Object 54
5.9 Supplementary Bill References Object 54
5.10 Transaction Types Object 54
5.11 Channel Types Object 55
5.12 Fees Object 55
6 Enumerations 56
6.1 ISO Currency Codes 56
6.2 Transaction Types 56
6.3 ID Types 56
6.4 Account Identifiers 57
6.5 ISO Country Codes 59
6.6 Delivery Method Type 59
6.7 Frequency Type 59
7 API Behaviour & Error Handling 61
7.1 Use of HTTP Response Codes 61
7.2 API Behavioural Model 62
7.2.1 Overview 62
7.2.2 Request State Object 62
7.2.3 API Sequence Flow Patterns 63
7.2.4 API Sequence Flow Diagrams 65
7.3 HTTP Status Responses and Error Categories 72
7.3.1 Client Error Categories 72
7.3.2 Server Error Categories 72
7.4 Error Codes Definition 73
7.5 Errors Object Definition 74
7.6 API Heartbeat 74
7.7 Missing Response Retrieval 75
7.8 Harmonised Error Codes 76

1 Introduction

The purpose of this document is to detail the design principles, objects, behaviours and error handling for the Mobile Money API.

The overriding goal of the API is to enable all parties to implement mobile money APIs in a flexible, yet consistent manner. This has been achieved by use of the following principles:

- Use of REST architectural principles.
- Providing a set of well-defined objects that are abstracted from the underlying object representations held in the various mobile money systems. This allows an API client to construct an API message without requiring specific knowledge of the target server implementation.
- Creation of a standard set of transaction types and other key enumerations, removing the need for developers to map for each and every API implementation.
- Use of ISO international standards for enumerators such as currency and country codes
- Use of supplementary metadata and sub-types to enable use case and/or mobile money provider-specific properties to be conveyed where necessary.
- Recognising that no common mobile money account identifier exists, use of a flexible construct to enable the target account(s) and transaction parties to be identified using one or multiple identifier types.

This documentation contains the following sections:

- **Quick Start Guide.** Provides a basic reference point to enable a developer to identify which objects and paths to use for their target use cases.
- **API Fundamentals**. The core principles and constructs that underpin the API.
- **API Service Definition**. Details the available API Services.
- **API Supporting Object Definition**. Details the properties for the supporting objects that constitute the harmonised API.
- **Enumerations**. Describes all of the list of values that are applied to specific object properties. Where international standards are used, appropriate references are supplied.
- **API Behaviour and Error Handling**. Describes behavioural aspects of the API and details error handling including error code definition, Heartbeat API definition and a detailed explanation of synchronous and asynchronous methods.

1.1 Definitions of Terms

Term	Description
ATM	Automated Teller Machine. Enables customer to be able to perform financial transactions without the need for a human cashier. For mobile money, the ATM machine can be considered as an automated agent, allowing withdrawals and some case deposits from and to the customer's wallet.

Agent	A person or business that is contracted to facilitate transactions for users. The most important of these are cash-in and cash-out (i.e. loading value into the mobile money system, and then converting it back to cash again); in many instances, agents register new customers too.
API Service	An API Service comprises of a URI which when combined with a HTTP method (GET, PATCH, POST) enables a specified operation to be conducted on a specific resource or set of resources.
Application Programming Interfaces (APIs)	In computer programming, an application programming interface (API) is a set of routines, protocols, and tools for building software applications. An API expresses a software component in terms of its operations, inputs, outputs, and underlying types.
Authentication	Confirms the identity of the entity that is requesting the service by verifying supplied credential(s). In this document, the requesting entity is typically referred to as the initiator or sender.
Authorisation	Authorisation provides an authenticated identity with access to the system resources/services to which they are entitled. In the context of this document, the requesting entity (initiator) is provided with a defined set of services and functions that they can perform over API.
Business	Describes an entity such as a public limited or limited company that uses mobile money as a service. This includes taking bill payments, making bill payments and disbursing salaries
Client	Describes the calling system/party that initiates and interacts with the API service. The Client may or may not be the debit or credit party
HTTP	Acronym for Hyper Text Transfer Protocol. HTTP is the foundation of data communication for the World Wide Web. HTTP is the protocol to exchange or transfer hypertext.
ISO	International Organisation for Standardisation. A number of ISO standards are used with the mobile money API.
ITU	International Telecommunications Agency. ITU MSISDN format standards have been adopted for the mobile money API.
JSON	Acronym for JavaScript Object Notation. Used to transmit data objects containing attribute/value pairs. One of two protocols (the other being XML) considered as the basis for a Data Exchange format for mobile money API harmonisation
Merchant	A retail merchant that accepts e-Money from mobile money customers in exchange for goods and/or services.
MSISDN	A number uniquely identifying a subscription in a GSM or a UMTS mobile network. Analogous with a mobile phone number.
Object	An object holds a set of properties. Objects can be combined or used in isolation to provide a representation of a resource.
Organisation	Describes a non-business entity such as a charity or government department that uses mobile money as a service. This includes taking bill payments, making bill payments and disbursing salaries
Provider	Hosts the Mobile Money API services (or subset of).
Resource	A resource is the target for an API operation as identified in the API request. Resources are conceptually different from the representations that are returned to the client.

REST-based API	In computing, Representational State Transfer (REST) is the software architectural style adopted by many organisations that offer APIs. REST stipulates that interacting systems exhibit the following characteristics: • **Client-Server**. Responsibilities between clients and servers are explicitly defined. • **Stateless**. No context is stored between client and server during communication. • **Cacheable**. As on the World Wide Web, clients and intermediaries can cache responses • **Layered System**. A client cannot ordinarily tell whether it is connected directly to the end server, or to an intermediary along the way. **Uniform Interface**. A uniform interface simplifies and decouples the architecture, which enables each part to evolve independently.
URI	Acronym for Uniform Resource Identifier. The URI is a string of characters used to identify a resource (such as an account, for example). In APIs a URI is represented in the form of a URL which is generally used to identify the service to invoke.
UUID	Universally Unique Identifier. UUIDs are usually stored as 128-bit values and are used for as the datatype for correlation IDs in the mobile money API.
Wallet	Term used in this document to describe the store of value that identities (customers, agents, merchants and businesses) use to store e-Money. In general, the document uses the term Account in place of Wallet.

2 API Quick Start Guide

I want to	Path	Refer to these sections
Make a Bill Payment	/accounts/bills/payments	Bill Payments API
Create or view a Disbursement	/transactions	Transactions APIs
Create or view a Deposit	/transactions	Transactions APIs
Create or view an International transfer	/transactions	Transactions APIs
Create or view a Merchant Payment	/transactions	Transactions APIs
Create or view an Adjustment (including refunds)	/transactions	Reversals API
Create or view a Reversal	/transactions	Reversals API
Create or view a Domestic Transfer	/transactions	Transactions APIs
Create or view a Withdrawal	/transactions	Transactions APIs
View an Account Status	/accounts/status	Accounts APIs Accounts Status API
View Balance(s) for an account	/accounts/balance	Accounts APIs Balance API
View an Account Holders Name	/accounts/accountname	Accounts APIs Account Name API
View Account Statement Entries for an account	/accounts/statemententries	Statement Entries API
View a specific Statement Entry	/statemententries	Statement Entries API
View Bill Payment Service Providers	/accounts/billcompanies	Bill Companies API
View Bills for presentment	/accounts/bills	Bills API
Create or update Debit Mandates	/accounts/debitmandates	Debit Mandates API
Create, view or cancel an Authorisation Code	/accounts/authorisationcodes	Authorisation Codes API
Create or view a Remittance Quotation	/quotations	Quotations API
Create, view or update an account to account Link	/accounts/links	Links API
Create a batch of transactions	/batchtransactions	Batch Transactions API
Retrieve batch transaction errors	/batchtransactions/rejections	Batch Transactions API
Retrieve batch transaction completions	/batchtransactions/completions	Batch Transactions API

3 API Fundamentals

3.1 URI

All services exposed by the harmonised Mobile Money API use the following URI format:

{…]/{version}/mm/{Resource}

Where:

- **…** is defined upon implementation of the API by the API provider.
- **version** is as per standards defined in the API Versioning section.
- **mm** is literal for 'Mobile Money'.
- **resource** identifies the object and resource that is the subject of the API.

3.2 Methods

The API supports three types of request methods, each represented by a HTTP method as shown below:

- **POST**. Used to create a resource for a given resource type. Transactions, Batch Transactions, Quotations, Bill Payments, Links, Authorisation Codes and Debit Mandates can be created.
- **PATCH**. Used to update a resource for a given resource type. Batch Transactions, Links, Authorisation Codes and Debit Mandates can be updated. Note that only specific properties can be updated – refer to the relevant API for more information.
- **GET**. Used to return a representation(s) of a resource(s) or a collection of resources.

3.3 Patch Specifics

Updates to resources are accomplished by use of the HTTP PATCH method. The PATCH format is based upon IETF RFC 6902. The **replace** operation is supported as per below:

Replace. This operation replaces the value of the target property with the supplied value. An example of a **replace** operation is ***[{ "op": "replace", "path": "/XYZ", "value": test }]*** where XYZ is the target property.

3.4 Resource Naming

The resource part of the URI path identifies the type of resource and if applicable, the specific resource for which an operation is to be performed. Resources are reflected in plural and by use of nouns. Primary resource types are as follows:

- ***/transactions***. Identifies a transaction resource.
- ***/accounts***. Identifies the financial resource that holds the funds. Examples include bank accounts and mobile wallets.
- ***/quotations***. The resource that holds the quote or quotes for domestic and international remittance quotations.
- ***/statemententries***. The resource that returns an account statement representation. This is typically qualified by /accounts but can also be used to return a specific statement entry if the transaction reference is known.

- ***/accounts/debitmandates***. The resource that holds the debit mandate instruction(s), i.e. mandates that permit a Payee to deduct funds from a Payer's account.
- ***/accounts/links***. Represents an account to account link between two systems.
- ***/accounts/authorisationcodes***. Represents an authorisation code which pre-authorises a transaction.
- ***/accounts/bills***. Represents a bill that is due to be paid or has been paid.
- ***/accounts/bills/payments***. Represents the payment of a bill.
- ***/accounts/accountname***. Represents the primary account holders name.
- ***/accounts/balance***. Represents the balance(s) associated with an account.
- ***/accounts/status***. Represents the ability of the account to receive/send funds.
- ***/batchtransactions***. Represents a collection of transactions and related batch metadata.

A more exhaustive list of permitted resources can be found in the detailed sections of this document.

3.5 Identifying the Resource

The harmonised Mobile Money API adopts a RESTful approach to identifying the resource on which an operation is to be performed. Where possible, the identifier is specified following the resource name in the path. In some cases, more than one identifier is required to identify the resource or there is a choice on which identifier to use. Therefore, flexibility has been built into the path to cater for these cases. This is illustrated in the following table.

Resource	Identifier	Identifier Placement
Transactions	Transaction Reference	Path
Accounts	Various account identifiers (see Accounts for details)	Path
Mandates	Mandate Reference	Path
Quotations	Quotation Reference	Path
Statement Entries	None if qualified by Accounts. If individual entry, then use Transaction Reference	Path
Links	Link Reference	Path
Bills	Account Identifiers	Path
Bill Payments	Bill Payment Reference and Account Identifiers	Path
Batch Transactions	Batch Reference	Path
Authorisation Codes	Authorisation Code	Path

3.6 Client Correlation ID

A Client Correlation ID can be supplied by the API client on HTTP POST and PATCH requests. The Client Correlation ID is a unique identifier that enables the client to correlate the API request with the resource created/updated by the provider. The client correlation ID is captured in the HTTP Header.

The format for the correlation ID is a UUID. Critically, the correlation ID supports safe operations. A POST request that is submitted with a correlation ID that has already been supplied will be rejected as unsafe, thus avoiding transaction duplication.

3.7 Use Case Flow Patterns

All Use Cases supported by the API are built on standardised flow patterns. Flow patterns exist for viewing, creating and updating records (resources). Flow patterns can be combined in a single use case. For example, a domestic transfer may involve a name check on the recipient's platform prior to a request to create a transaction. There are some simple rules to follow when using flow patterns. These are illustrated below:

- **Viewing**. Only synchronous flows are supported for viewing resource(s). For example, a HTTP GET request to view balances must result in a HTTP response yielding the representation of the resource or the representation of the error.
- **Creating/Updating.** Synchronous and asynchronous flows are supported. A synchronous flow involves a definitive result consisting of the resource representation or error representation being returned in the GET, POST or PATCH response. An asynchronous flow is supported by two mechanisms – Call-back and Polling. These are described in more detail in the API Behavioural Model section.

3.8 Case Sensitivity

All API properties are defined in camelCase format.

All enumeration values referenced within the API use lower case notation – this includes acronyms and abbreviations. The only exceptions are for

- ISO Codes (country and currency) – the API uses these codes as defined per ISO.
- Error Codes. Upper Case is used to identify the first letter of each word to assist readability.

3.9 HTTP Header Information

The following header information can be supplied for the mobile money API.

For the security headers, please also refer to the Mobile Money API Security Design for further information.

3.9.1 Standard Request Headers

Header	Value	Optionality	Notes
Accept	application/json	Mandatory	
Accept-Charset	utf-8	Mandatory	
Authorization	Authorization: Basic {base64Encode(concatenated client's username followed by ':' and password)} OR OAuth2 Access Token. For OAuth2 format is {'Bearer' *token value*}	Mandatory	

Content-Length	Length of request content in 8-bit bytes	Mandatory	
Content-Type	application/json	Mandatory	

3.9.2 Standard Response Headers

Header	Value	Optionality	Notes
Content-Length	Length of response content	Conditional	Applicable only if the HTTP response contains JSON body
Content-Type	application/json; charset=utf-8	Conditional	Applicable only if the HTTP response contains JSON body

3.9.3 Custom Request Headers

Header	Value	Optionality	Notes
X-API-Key	Used to pass pre-shared client's API key to the server	Conditional	Only required when API Client Authentication based on API key is used.
X-User-Bearer	Used to pass user's access token	Conditional	Only required when OAuth 2.0/OIDC authorisation framework is used for end-user authentication.
X-Date	{The date and time that the message was sent in HTTP-date format - RFC 7231}	Conditional	Used for Basic message integrity checks.
X-Client-Id	Used to pass pre-shared client's identifier to the server	Conditional	Can be used in addition to X-API-Key.
X-Content-Hash	SHA-256 hex digest of the request content (encrypted or plain)	Conditional	Applicable only if the HTTP request contains JSON body and basic data integrity checking is to be performed.
X-CorrelationID	UUID	Conditional	Please refer to Client Correlation ID.
X-User-Credential-1	Contains an authentication credential of the end user (e.g. PIN, Password).	Conditional	Should only be used when OAuth 2.0/OIDC authorisation framework has not been implemented by the API Provider.
X-User-Credential-2	Contains an authentication credential of the end user (e.g. PIN, Password). Can be used when a second credential is required.	Conditional	Should only be used when OAuth 2.0/OIDC authorisation framework has not been implemented by the API Provider.

X-Channel	string containing the channel that was used to originate the request. For example, USSD, Web, App..	Conditional	Used to identify the API invocation channel.
X-Callback-URL	string containing the URL which should receive the Callback for asynchronous requests.	Conditional	Will only be used by the API provider if they have implemented the Callback method.
X-Account-Holding-Institution-Identifier-Type	A header variable that identifies the type of the account holding institution. This header is used to support request routing and should be used in conjunction with the X-Account-Holding-Institution-Identifier header.	Optional	Supports three enum values: • lei • swiftbic • organisationid If swiftbic or lei are specified, please ensure that the X-Account-Holding-Institution-Identifier contains a valid value for these types.
X-Account-Holding-Institution-Identifier	A header variable that identifies the account holding institution. This header is used to support request routing and should be used in conjunction with the X-Account-Holding-Institution-Identifier-Type header.	Optional	When lei is supplied in X-Account-Holding-Institution-Identifier-Type, please refer to https://www.leiroc.org/lei.htm for permissible values. When swiftbic is supplied in X-Account-Holding-Institution-Identifier-Type, please refer to https://www.swift.com/standards/data-standards/bic for permissible values.

3.9.4 Custom Response Headers

Header	Value	Optionality	Notes
X-Records-Available-Count	integer containing number of records that are available to be returned	Conditional	Used in the event of pagination
X-Records-Returned-Count	integer containing the number of records that have been returned	Conditional	Used in the event of pagination
X-Date	{The date and time that the response was	Conditional	Used for Basic message integrity checks

	sent in HTTP-date format - RFC 7231}		

3.10 API Versioning

When changes are made to the Mobile Money API, a new version is released. There are three types of API versions:

- major (backwards incompatible)
- minor (backwards compatible)
- patch (backwards compatible)

The following types of changes are considered to be backwards compatible:

- Addition of new API Services.
- Adding optional request properties and/or optional input parameters such as query strings to existing objects.
- Addition of new properties to existing API responses.
- Changing the order of properties within a request or response object.
- New error codes.

The following types of changes are considered to be backwards incompatible and hence major:

- Introducing mandatory properties.
- Changing datatypes on properties.
- Changes to API URIs/Paths.

The version that a client intends to use is indicated in the path. Format is ‘X.Y.Z’ where ‘X’ is the major version, ‘Y’ is the minor version and ‘Z’ is a patch version. Versions are sequentially numbered. When a major version is incremented, the minor version is reset to zero.

The following table provides examples of API version compatibility:

Client Version	Provider Version	Compatible?
1.0.0	1.1.0	Yes
1.1.0	1.0.0	Client would need to submit only 1.0.0 related payload. If 1.1.0 services, properties or parameters are supplied by the client, the provider will reject the request.
2.0.0	1.0.0	No
1.0.0	2.0.0	No

3.11 Amount Validation

The mobile money API applies common validation to all amount properties. The following rules are applied during validation:

- Between zero and four decimal places can be supplied.
- Leading zeroes are not permitted except where the value is less than 1. For any value less than one, one and only one leading zero must be supplied.
- Trailing zeroes are permitted.
- Negative values are not permitted.
- Maximum value that can be supplied is 99999999999999999.9999.

Amount validation examples are shown below.

Value	Permitted?
5	Yes
5.0	Yes
5.	No
5.00	Yes
5.5	Yes
5.50	Yes
5.5555	Yes
5.55555	No
555555555555555555	Yes
5555555555555555555	No
-5.5	No
0.5	Yes
.5	No
00.5	No
0	Yes
00.00	No
0.00	Yes
0000001.32	No

4 API Service Definition

All Mobile Money API Services and related objects are detailed within the API Service Definition and Supporting Objects sections. The relationships between objects is illustrated in figure 1.

Note: That green represents objects that are directly exposed by an API service and blue represents objects that are referenced by directly exposed objects.

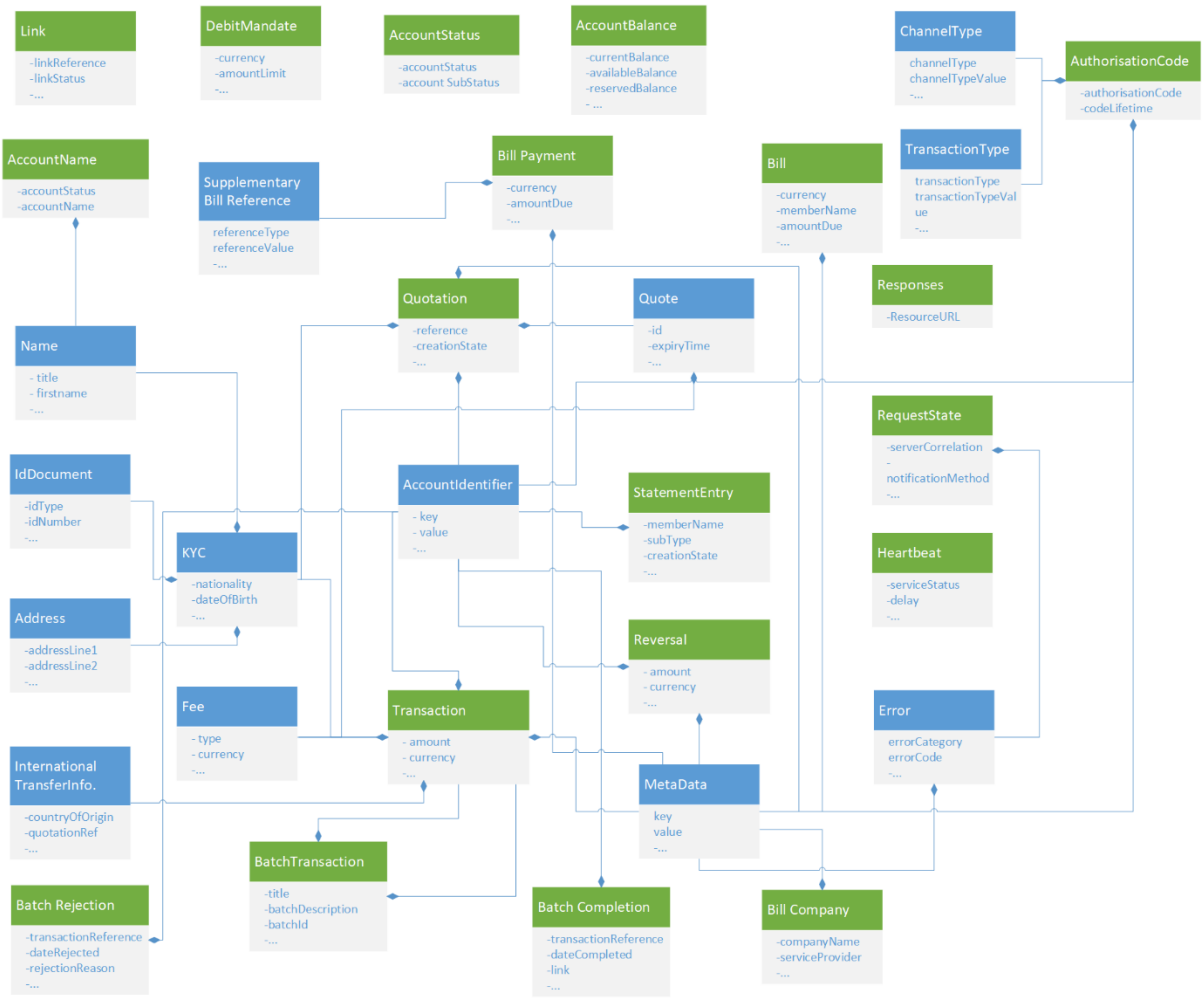

Figure 1. API Object Relationships

Object properties are described in this specification as follows:

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

  → Request optionality

  ← Response optionality

O Property is optional

M Property is mandatory

C Property is conditional

NA Property does not need to be supplied. If supplied, it will be ignored.

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

Note: All string properties have a default maximum length of 256 characters unless specified otherwise.

4.1 Transactions API

The Transactions APIs are used to support mobile money financial transaction use cases. Transactions are used for a wide range of use cases including merchant payments, international transfers, domestic transfers and agent cash-in/cash-out. Reversals and adjustments are also treated as transactions.

The following paths are permitted:

- ***POST /transactions***. For transaction creation.
- ***POST /transactions/type/{transactiontype}***. To be used for transaction creation when the provider's API Gateway requires that the transaction type is identified in the URL. When this path is used, the Transaction Type does not need to be supplied in the request body.
- ***GET /transactions/{transactionReference}***. To view a specific transaction.

The object definition for Transactions is provided below:

Transaction Base Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
amount	string	The transaction amount.	→M ←M		Please refer to Amount Validation
currency	string	Currency of the transaction amount.	→M ←M		Enumeration = ISO Currency Codes
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		
descriptionText	string	Free format text description of the transaction provided by the client. This can be provided as a reference for the	→O ←O		

		receiver on a notification SMS and on an account statement.			
requestDate	date-time	The creation date and time of the transaction as supplied by the client.	→O ←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		
transactionReference	string	Unique reference for the transaction. This is returned in the response by API provider.	→NA ←M		
transactionReceipt	string	Transaction receipt number as notified to the parties. This may differ from the Transaction Reference.	→NA ←O		
requestingOrganisationTransactionReference	string	A reference provided by the requesting organisation that is to be associated with the transaction.	→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 property for requests that have been pre-authorised.	→O ←O		
geoCode	string	Indicates the geographic location from where the transaction was initiated.	→O ←O		
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.
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.
senderKyc	object	A collection of properties detailing the KYC of the transaction Sender, typically used for International Transfers.	→O ←O	KYC Information	
recipientKyc	object	A collection of properties detailed the KYC of the transaction Recipient, typically used for International Transfers.	→O ←O	KYC Information	
internationalTransferInformation	object	A collection of properties detailing information specifically used for international transfers.	→O ←O	International Transfer Information	
originalTransactionReference	string	For reversals and refunds, this property indicates the transaction which is the subject of the reversal.	→O ←O		
servicingIdentity	string	The property is used to identify the servicing identity for transactions, e.g. till, POS ID, assistant ID.	→O ←O		

requestingLei	string	Legal Entity Identifier of the organisation that is requesting the transaction.	→O ←O		Refer to LEI format as defined here: https://www.leiroc.org/lei.htm
receivingLei	string	Legal Entity Identifier of the organisation that is receiving the transaction.	→O ←O		Refer to LEI format as defined here: https://www.leiroc.org/lei.htm
fees	array	Allows the passing and/or returning of all fees pertaining to the transaction.	→O ←O	Fees Object	
metadata	array	A collection of key/value pairs. These can be used to populate additional transaction properties.	→O ←O	Metadata	

4.2 Reversals API

The Reversals API is used to reverse or adjust/refund a financial transaction. The originating transaction reference must be provided in the path in order to identify the transaction to be reversed. For a partial reversal, the amount needs to be supplied. It should be noted that some API Providers do not support partial reversals and will return an error if a partial amount is supplied.

For viewing reversals, the Transactions API should be used.

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

The object definition for Reversals is provided below.

Reversal Base Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
amount	string	The transaction Amount.	→O ←O		Please refer to Amount Validation
currency	string	Currency of the transaction amount.	→O ←O		Enumeration = ISO Currency Codes.
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		
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		
requestDate	date-time	The creation date and time of the transaction as supplied by the client.	→O ←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		
transaction Reference	string	Unique reference for the transaction. This is returned in the response by API provider.	→NA ←M		
transaction Receipt	string	Transaction receipt number as notified to the parties. This	→NA ←O		

		may differ from the Transaction Reference.			
geoCode	string	Indicates the geographic location from where the transaction was initiated.	→O ←O		
requesting OrganisationTransactionReference	string	A reference provided by the requesting organisation that is to be associated with the transaction.	→O ←O		
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	
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	
originalTransactionReference	string	For reversals and refunds, this property indicates the transaction which is the subject of the reversal.	→NA ←M		
requesting Lei	string	Legal Entity Identifier of the organisation that is requesting the transaction.	→O ←O		Refer to LEI format as defined here: https://www.leiroc.org/lei.htm
receiving Lei	string	Legal Entity Identifier of the organisation that is receiving the transaction.	→O ←O		Refer to LEI format as defined here: https://www.leiroc.org/lei.htm
servicingIdentity	string	The property is used to identify the servicing identity for	→O ←O		

		transactions, e.g. till, POS ID, assistant ID.			
fees	array	Allows the passing and/or returning of all fees pertaining to the transaction.	→O ←O	Fees Object	
metadata	array	A collection of key/value pairs. These can be used to populate additional transaction properties.	→O ←O	Metadata	

4.3 Introduction to Processing Batch Transactions

The Mobile Money API allows clients to submit, approve and view batches of transactions. The following steps describe an end to end flow for processing batch transactions. 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 individual APIs that are referenced in the steps below are fully documented in subsequent sub-sections.

4.3.1 One-Shot Batch Processing

Creating a Batch

1. Client submits the batch for processing via a '***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 '**completed**'.

Verifying a Batch

5. 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.
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**'.

4.3.2 Batch Processing with Maker/Checker

Creating a Batch

1. Client submits the batch for processing via a '***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 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.
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***'.

4.4 Batch Transactions API

This API enables clients to submit and approve a batch of transactions. The API allows transactions of multiple types to be include in a single batch. The following methods and 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.

A limit of 999,999 transaction records per batch has been implemented in the Swagger definition.

Batch Transaction Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
batchTitle	string	Client-provided title for the batch.	→O ←O		
batchDescription	string	Client-provided description of the batch.	→O ←O		
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
processingFlag	Boolean	Indicates whether the batch is currently undergoing processing by the API Provider.	→N/A ←O		
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 ←M		
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		

rejectionCount	integer	Indicates the number of records that have been rejected, either during parsing or during final processing.	→NA ←O		
parsingSuccessCount	integer	Indicates the number of records that have been parsed successfully.	→NA ←O		
completedCount	integer	Indicates the number of records that have been successful completed.	→NA ←O		
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	

4.5 Batch Rejections API

This API enables clients 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 date for which records should be returned.
toDateTime	string	date-time	Indicates the maximum date 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).

Batch Rejection Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
transaction Reference	string	Transaction Reference as assigned by the API provider.	→N/A ←O		
rejectionDate	date-time	Date and time of the rejection.	→N/A ←M		
debitParty	array	The debit party identifiers for the transaction as specific in the batch request.	→N/A ←M	Account Identifiers	
creditParty	array	The credit 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		
requesting OrganisationTransactionReference	string	A reference provider by the requesting organisation that is to be associated with the transactions.	→N/A ←O		

4.6 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 date for which records should be returned.
toDateTime	string	date-time	Indicates the maximum date 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)..

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

4.7 Introduction to Accounts APIs

The Accounts APIs are used to support a range of operations on a financial account resource and associated resources. Types of accounts include mobile wallets, financial institution accounts and utility accounts (e.g. for electricity).

4.7.1 Identifying a Target Account

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

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}***.

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 list of permitted account identifiers supported by the Mobile Money API can be found in the Account Identifiers section. As there can be multiple identifiers required to identify the target account, the path uses a '$' delimiter to separate each identifier, up to a limit of three account identifiers.

4.7.2 Supported Account Resources

The Accounts API support operations against accounts and related resources as per below:

- ***/accounts/status***. Returns the current status for an account. See the Account Status API for more information.
- ***/accounts/accountname***. Returns all name properties held for the primary identity that is associated with the account. See the Account Name API for more information.
- ***/accounts/balance***. Returns the balances for the account. See the Account Balances API for more information.
- ***/accounts/statemententries*.** Returns all statement entries for a given account. See the Statement Entries API for more information.
- ***/accounts/bills*.** Returns all outstanding bills for a given account and enables bills to be paid. See the Bills API and Bill Payments API for more information.
- ***/accounts/billcompanies*.** Returns all applicable bill companies for given account. See the Bill Companies API for more information.
- ***/accounts/debitmandates*.** Allows the creation, updating and viewing of debit mandates for a given account. See Debit Mandates API for more information.
- ***/accounts/links*.** Allows the creation, updating and viewing of account to account links for a given account. See Links API for more information.
- ***/accounts/authorisationcodes*.** Allows the creation, updating and viewing of authorisation codes for a given account. See Authorisation Codes API for more information.

4.8 Account Transactions API

It is possible to return a range of transactions for an account as per the following methods and paths:

GET /accounts/{identifierType}/{identifier}/transactions or

GET /accounts/{Account Identifiers}/transactions.

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 date for which records should be returned.
toDateTime	string	date-time	Indicates the maximum date 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)..

4.9 Accounts Status API

The Accounts Status API returns a harmonised status of the account. The status enables the client to determine whether transactions can be subsequently posted against the account. Permitted methods and paths are ***'GET /accounts/{identifierType}/{identifier}/status'*** or ***'GET /accounts/{Account Identifiers}/status'***.

Account Status Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
accountStatus	string	Indicates a simplified representation of the account status. This will be shown as 'available' or 'available'. A state of 'unavailable' means that the	→NA ←M		Enumeration = available, unavailable, unregistered

		account is in a state that does not allow posting of transactions. Unregistered indicates that although not available, a transaction posted with the account identifier(s) will result in an unregistered voucher creation.			
subStatus	string	Property can be used to return a provider-specific status for the account.	→NA ←O		
Lei	string	Indicates the Legal Entity Identifier of the organisation holding the account.	→NA ←O		Refer to LEI format as defined here: https://www.leiroc.org/lei.htm

4.10 Account Balances API

This API defines specific properties for returning balances associated with an account. Permitted methods and paths are '***GET /accounts/{identifierType}/{identifier}/balance***' or '***GET /accounts/{Account Identifiers}/balance***'.

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

Balance Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
currentBalance	string	The current outstanding balance on the account.	→NA ←O		Please refer to Amount Validation
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 Amount Validation
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 Amount Validation
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 Amount Validation
currency	string	Currency for all returned balances.	→NA ←O		Enumeration = ISO Currency Codes
accountStatus	string	Indicates a harmonises representation of the account state. This will be shown as	→NA ←O		Enumeration = available,

		'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.			unavailable, unregistered

4.11 Account Holder Name API

This API defines specific properties for returning account holder name information associated with an account. Permitted methods and paths are '***GET /accounts/{identifierType}/{identifier}/accountname***' or '***GET /accounts/{Account Identifiers}/accountname***'.

Account Name Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
name	Reference	A collection of properties detailing the name of the Primary Account Holder.	→NA ←O	Name	
lei	string	Indicates the Legal Entity Identifier of the organisation holding the account.	→NA ←O		Refer to LEI format as defined here: https://www.leiroc.org/lei.htm

4.12 Statement Entries API

The Statement Entries API enables generic representations of transactions to be returned. Typically, the returned representations are used for the purposes of presenting a statement to the account holder. In order to return statement entries, an account or a transaction must be specified. The supported methods and paths are as follows:

To return a specific statement entry:

- ***GET /statemententries/{transactionReference}***

To return a range of statement entries:

- ***GET /accounts/{ identifierType}/{identifier}statemententries*** or ***/accounts/{Account Identifiers}/statemententries***.

To filter the number of records returned, the following query string parameters 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 date for which records should be returned.
toDateTime	string	date-time	Indicates the maximum date for which records should be returned.
transactionStatus	string	N/A	Indicates the status of the transactions to be returned.
displayType	string	N/A	Indicates the Display Type of the transactions to be returned.

Note 1: For a harmonised behavior, API Providers should make sure that the statement entries 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).

Statement Entries Object Properties

Name	Type	Description		Reference	Validation
amount	string	Amount of the transaction.	→NA ←M		Please refer to Amount Validation
currency	string	Currency of the transaction.	→NA ←M		Enumeration = ISO Currency Codes
displayType	string	The transaction type that is to be used for presentation to the account holder as determined by the API provider. This is not necessarily the actual transaction type.	→NA ←O		
transactionStatus	string	Indicates the status of the transaction as represented by the API provider.	→NA ←M		
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.	→NA ←O		
requestDate	date-time	The creation date and time of the transaction as supplied by the client.	→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 modified by the API Provider.	→NA ←O		
transaction Reference	string	Unique reference for the transaction. This is returned in the response by API provider.	→NA ←M		
transaction Receipt	string	Transaction receipt number as notified to the parties. This may differ from the Transaction Reference.	→NA ←O		
debitParty	array	A collection of key/value pairs that identify the debit. Keys include MSISDN and Wallet Identifier.	→NA ←M	Account Identifiers	
creditParty	array	A series of key/value pairs that identify the credit party. Keys include MSISDN and Wallet Identifier.	→NA ←M	Account Identifiers	

4.13 Bills API

The Bills API is used to return all outstanding bills associated with an account. The main purpose of the API is to support Bill Presentment, i.e. presenting all applicable bills for a payer to view and select for payment. In order to pay a bill, the Bill Payments API is used. Permitted methods and paths are '***GET /accounts/{identifierType}/{identifier}/bills***' or '***GET /accounts/*{Account Identifiers}*/bills***'.

Bill Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
currency	string	Currency of the bill to be paid.	→NA ←O		Enumeration = ISO Currency Codes
amountDue	string	Amount outstanding on the bill to be paid.	→NA ←O		Please refer to Amount Validation
dueDate	date	Date on which the Bill is due to be paid.	→NA ←O		
billReference	string	Reference number for the Bill that the payer can use when making a payment.	→NA ←O		
minimumAmountDue	string	The minimum amount that is outstanding on the bill to be paid.	→NA ←O		Please refer to Amount Validation

billDescription	string	Description of the bill that is to be paid.	→NA ←O		
billStatus	String	Identifies the status of the Bill.	→NA ←O		‘paid’, ‘unpaid’, ‘partialpaid’
metadata	array	A collection of key/value pairs. These can be used to return additional information regarding the bill.	→NA ←O	Metadata	

4.14 Bills Payments API

The Bills Payments API is used to pay a specific bill associated with an account. Permitted methods and paths are ‘***POST /accounts/{identifierType}/{identifier}/bills/{billReference}/payments***’ or ‘***POST /accounts/*{Account Identifiers}*/bill/{billReference}/payments***’

Both synchronous and asynchronous modes are supported.

Bill Payment Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
currency	string	Currency of the amount that is being paid.	→M ←M		Enumeration = ISO Currency Codes
amountPaid	string	Amount that is being paid.	→M ←M		Please refer to Amount Validation
billPaymentStatus	String	Indicates the status of the bill payment as stored by the API provider.	→N/A ←M		
serviceProviderPaymentReference	String	Reference for the payment generated by the service provider.	→O ←O		
requestingOrganisation	String	The originating mobile money provider or financial institution that holds the wallet/account of the payer.	→O ←O		
requestingOrganisationTransactionReference	String	The mobile money provider’s (or Financial Institution’s) transaction reference used to debit the customer and credit the service provider.	→O ←O		

customerReference	string	Textual reference provided by the customer paying the bill.	→O ←O		
paymentType	string	Describes the type of Bill Payment, i.e. whether a full or partial payment.	→O ←O		Enumeration = 'fullpayment', 'partialpayment'
supplementaryBillReferenceDetails	array	In some cases, a single reference is not sufficient to identify a bill. This key-value collection enables further reference information to be supplied.	→O ←O	Supplementary Bill References	
metadata	Array	A collection of key/value pairs. These can be used to populate additional bill payment properties.	→O ←O	Metadata	
serviceProviderComment	String	Allows the Service Provider to include specific information regarding the bill payment.	→N/A ←O		
serviceProviderNotification	String	Allows the Service Provider to include specific information that will be included on the notification to the customer by the mobile money provider.	→N/A ←O		

4.15 Bill Companies API

The Bill Companies API is used to return a list of Service Providers that accept Bill Payments. Permitted methods and paths are as per below:

- Bill Companies irrespective of account:
  - Use '***GET /billcompanies'*** to return a list of all bill payment service providers.
  - Use '***GET /billcompanies/{serviceProvider}***' to return a specific bill payment service provider.
- Bill Companies for a given account:
  - '***GET /accounts/{identifierType}/{identifier}/billcompanies***' or '***GET /accounts/{Account Identifiers}/billcompanies***'.

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 may apply a limit of 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.

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).

Bill Companies Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
companyName	String	Display Name for the Service Provider.	→NA ←M		
serviceProvider	String	Service Provider Reference Code.	→NA ←M		
service ProviderType	String	Type of Service Provider that accepts payments.	→NA ←O		
serviceProviderSubType	String	Sub-Type of Service Provider.	→NA ←O		
supplementary ServiceProviderDetails	Array	In some cases, further information for a service provider can be returned. This key-value collection enables further information to be supplied.	→NA ←O	Supplementary Service Provider Details	

4.16 Debit Mandates API

The Debit Mandates API is used to enable a mobile money customer to provide prior approval for payments to be taken from their account by the indicated payee. If the amount property is not supplied, the mandate is considered open, i.e. the payee would be able to take any amount. Mandates can be created, changed and inactivated. The permitted methods and paths as follows:

- Creation: ***POST /accounts/{identifierType}/{identifier}/debitmandates*** or ***POST /accounts/{Account Identifiers}/*****debitmandates**.
- Update: In order to update a debit mandate, a HTTP PATCH is used. Format is: ***PATCH /accounts/{identifierType}/{identifier}/debitmandates/{mandateReference}*** or ***PATCH /accounts/{Account Identifiers}/debitmandates/{mandateReference}***
- Read. ***GET /accounts/{identifierType}/{identifier}/debitmandates/{mandateReference}*** or ***GET /accounts/{Account Identifiers}/debitmandates/{mandateReference}***.

Synchronous and asynchronous modes are supported for the POST and PATCH methods whereas only synchronous mode is supported for the GET method.

Debit Mandate Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
currency	string	Currency of the amount limit.	→O ←O		Enumeration = ISO Currency Codes
amountLimit	string	The maximum amount that can be taken by the Payee on a payment request.	→O ←O		Please refer to Amount Validation
startDate	date	Date on which the mandate starts. If a frequencyType is specified, this will also be the date on which the first payment is to be taken.	→M ←M		
endDate	date	Date on which the Debit Mandate ends.	→O ←O		
numberOfPayments	number	Indicates the number of consecutive payments that are to be taken.	→O ←O		
frequencyType	string	Indicates the frequency for which payments will be taken from the payers account.	→O ←O		Enumeration = Frequency
mandateStatus	string	Indicates the status of the Debit Mandate as held in the API Provider system.	→O ←O		Enumeration = active, inactive
mandateReference	string	Unique reference provided by the API Provider for the Debit Mandate.	→N/A ←M		
requestDate	date-time	The creation date and time of the Debit Mandate as supplied by the client.	→M ←M		
creationDate	date-time	Date and time when the Debit Mandate was created by the API Provider.	→NA ←O		

modificationDate	date-time	Date and time when the Debit Mandate was modified by the API Provider.	→NA ←O		

4.17 Links API

The Links API is used to establish a link between two separate accounts on the client and provider systems. The API can be used for example to link a mobile wallet account to a Microfinance Institution account or a bank account. The link object does not mandate the processes to verify and authenticate a link request - this depends upon the use case. A link needs to be associated with a mode of operation:

- pull. The link can be used by the client to debit the target account held by the provider.
- push. The link can be used by the client to credit the target account held by the provider.
- both. The link can be used for Push and Pull requests.

To identify the accounts that are to be linked, the target account is specified in the path whereas the source account is specified in the link object.

The permitted methods and paths are as follows:

- Creation: **POST** ***/accounts/{identifierType}/{identifier}/links*** or ***POST /accounts/{Account Identifiers}/links***.
- Update of status and/or mode properties: ***PATCH /accounts/{identifierType}/{identifier}/links/{linkReference}*** or ***PATCH /accounts/{Account Identifiers}/links/{linkReference}***.
- Read. ***GET /accounts/{identifierType}/{identifier}/links/{linkReference}*** or ***GET /accounts/{Account Identifiers}/links/{linkReference}***.

Synchronous and asynchronous modes are supported for POST and PATCH methods whereas only synchronous mode is supported for the GET method.

Link Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
linkReference	string	Indicates the Link reference. This enables a linked account to be uniquely identified.	→NA ←M		
status	string	Indicates the status of the Link.	→M ←M		Enumeration = active, inactive
mode	string	Indicates the mode of operation for the Link.	→M ←M		Enumeration = push, pull, both

sourceAccountIdentifiers	array	A series of key/value pairs that identify the source account. Keys include MSISDN and Wallet Identifier.	→M ←M	Account Identifiers	

4.18 Quotations API

The Quotations API are used to obtain one or multiple quotes for a mobile money customer that wishes to transfer money. The creation of a quote typically involves returning any fees that will be levied on the sending customer and if the request is international, the forex rate. A request is made for a quotation by the requesting Service Provider in response to a customer request. The quotation is calculated and returned. If the customer is satisfied with the quotation, then they can confirm and proceed with a transaction request using the /transactions API.

The following methods paths are permitted:

- Creation of a quotation: ***POST /quotations***
- View a quotation: ***GET /quotations/{Quotation Reference}***

Quotation Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
quotationReference	string	Unique reference for the Quotation as provided by the API Provider.	→NA ←M		
quotationStatus	string	Indicates the creation state of the Quotation.	→NA ←O		Enumeration = pending, rejected, completed
requestDate	date-time	The creation date and time of the Quotation as supplied by the client.	→M ←M		
creationDate	date-time	Date and time when the Quotation was created by the API Provider.	→NA ←O		
modificationDate	date-time	Date and time when the Quotation was modified by the API Provider.	→NA ←O		
debitParty	array	A collection of key/value pairs that enable the debit party to be identified. Keys include MSISDN and Wallet Identifier.	→M ←M	Account Identifier	

creditParty	array	A series of key/value pairs that enable the credit party to be identified. Keys include MSISDN and Wallet Identifier.	→M ←M	Account Identifier	
senderKyc	object	A collection of properties detailing the KYC of the transaction Sender, typically used for International Transfers.	→O ←O	KYC Information	
recipientKyc	object	A collection of properties detailing the KYC of the transaction Recipient, typically used for International Transfers.	→O ←O	KYC Information	
requestAmount	string	Requested Quotation amount.	→M ←M		Please refer to Amount Validation
requestCurrency	string	Currency of the requested Quotation amount.	→M ←M		Enumeration = ISO Currency Codes
type	string	The transaction type that the Quotation has been requested for.	→O ←O		Enumeration = Transaction Types
subtype	string	The transaction sub-type that the Quotation has been requested for.	→O ←O		
chosenDeliveryMethod	string	The delivery method chosen by the sending end user as the specific delivery method to be used in the quotes received.	→O ←O		Enumeration = Delivery Method
availableDeliveryMethod	string	Delivery Method that is possible for the intended recipient.	→NA ←O		Enumeration = Delivery Method
quotes	array	A collection of quotes. A quote can be received from a single receiving payment service provider or	→NA ←O	Quotes	

		from multiple providers.			
senderBlockingReason	string	The reason for blocking the Quotation, based on AML checks on the sender.	→N/A ←O		
recipientBlockingReason	string	The reason for blocking the Quotation, based on AML checks on the recipient.	→N/A ←O		
metadata	array	A collection of key/value pairs. These can be used to populate additional Quotation properties.	→O ←O	Metadata	

4.19 Authorisation Codes API

The Authorisation Codes API allows a mobile money payer or payee to generate a code which when presented to the other party, can be redeemed for an amount set by the payer or payee, depending upon the use case. Authorisation Codes are used widely in the industry across a range of use cases, including:

- ATM Codes for card-less withdrawals. A code is generated in advance by the customer and entered into the ATM to facilitate the withdrawal.
- Pre-authorisation codes for agent withdrawals. Depending upon the use case, this could involve the agent (payee) generating a code or the customer (payer) generating the code.
- Pre-authorisation codes for merchant payments. Depending upon the use case, this could involve the merchant (payee) generating a code or the customer (payer) generating the code.

Once an authorisation code has been generated, it can be presented through multiple means, including encoding into a QR code. Typically, an authorisation code will expire.

The following methods and 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}***.

Synchronous and asynchronous modes are supported for the POST and PATCH methods whereas only synchronous mode is supported for the GET method.

Authorisation Codes Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
authorisationCode	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’

requestDate	date-time	The date and time of the request.	→M ←O		
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		Must be positive value.
amount	string	Indicates the amount associated with the authorisation code. Typically, this is set by the client.	→O ←O		Please refer to Amount Validation
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'
holdFundsIndicator	Boolean	Indicates whether funds should be reserved against the payer's account where the payer is the requestor.	→O ←O		
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	
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	
metadata	array	A collection of key/value pairs. These can be used to populate additional properties.	→O ←O	Metadata	

5 Supporting Objects

5.1 International Transfer Information Object

The International Transfer Information object contains details that are specific to international transfers.

International Transfer Information Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
originCountry	string	The originating country of the transaction, i.e. the country where the transaction was initiated.	→M ←M		Enumeration = ISO Country Codes
quotationReference	string	Reference for the quotation that was provided to the sender. (refer to Quotations API for more information).	→O ←O		
quoteId	string	The specific quote associated with the quotation (refer to Quotes object for more information).	→O ←O		
receivingCountry	string	Destination Country of the international remittance.	→O ←O		
remittancePurpose	string	Property providing a description of the reason for the international remittance.	→O ←O		
relationshipSender	string	Indicates the relationship (if any) between the sender and the receiver.	→O ←O		
deliveryMethod	string	The recipient delivery method as chosen by the sender.	→O ←O		Enumeration = Delivery Method Types
senderBlockingReason	string	The reason for blocking the transaction, based on AML checks on the sender.	→NA ←O		
recipientBlockingReason	string	The reason for blocking the transaction, based on AML checks on the recipient.	→NA ←O		

5.2 KYC Information Object

KYC refers to 'Know your Customer'. The KYC object contains a number of properties that enable the identity of subject to be verified. KYC is typically provided for international transfers for the sending identity and the receiving identity. There are no mandatory KYC object properties.

KYC Information Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
nationality	string	Nationality of the KYC subject.	→O ←O		Enumeration = ISO Country Codes
dateOfBirth	date	Birth date of the KYC subject.	→O ←O		
occupation	string	Occupation of the KYC subject.	→O ←O		
employerName	string	Employer name of the KYC subject.	→O ←O		
contactPhone	string	Contact phone number (mobile or landline) of the KYC subject.	→O ←O		Must contain between 6 and 15 consecutive digits First character can contain a '+' or digit Can contain spaces.
gender	string	Gender of the KYC Object.	→O ←O		Length=1, Enumeration = (m)ale, (f)emale, (u)nspecified
idDocument	array	An array of properties containing the forms of identification that are associated with the subject.	→O ←O	Id Document	
postalAddress	object	A collection of properties that details the postal address of the KYC subject.	→O ←O	Address	
subjectName	object	Refers to the name properties for the KYC subject.	→O ←O	Name	
emailAddress	string	Email address of the KYC subject.	→O ←O		
birthCountry	string	The country of birth of the KYC subject.	→O ←O		Enumeration = ISO Country Codes

5.3 Name Object

The name object identifies the name details for the subject identity.

Name Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
title	string	The given title of the KYC subject, e.g. Mr, Mrs, Dr.	→O ←O		
firstName	string	First name (also referred to as given name) of the KYC subject.	→O ←O		
middleName	string	Middle Name of the KYC subject.	→O ←O		
lastName	string	Surname (also referred to as last or family name) of the KYC subject.	→O ←O		
fullName	string	The full name of the KYC subject.	→O ←O		
nativeName	string	The full name expressed as in the native language.	→O ←O		

5.4 ID Document Object

As part of KYC information, identification documentation is normally required. The ID Document Object enables documents pertaining to a subject's identity to be described.

ID Document Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
idType	string	Indicates the type of identification for the KYC subject, e.g. passport, driving licence etc..	→M ←M		Enumeration = ID Types
idNumber	string	Reference pertaining to the type of identification for the KYC subject.	→O ←O		
issueDate	date	Date of issue for the identification document.	→O ←O		
expiryDate	date	Date of expiry for the identification document.	→O ←O		

issuer	string	Indicates the organisation/government entity that issued the ID document.	→O ←O		
issuerPlace	string	Place of issue for the identification type.	→O ←O		
issuerCountry	string	Country where the identification type was issued.	→O ←O		Enumeration = ISO Country Codes
otherIdDescription	string	Where an ID Type of 'otherid' is specified, a description of the type of identification can be provided in this property.	→O ←O		

5.5 Address Object

The address object holds the postal address of the subject. Due to variability of address information in a number of mobile money markets, only Country is mandatory.

Address Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
addressLine1	string	First line of the address.	→O ←O		
addressLine2	string	Second line of the address.	→O ←O		
addressLine3	string	Third line of the address.	→O ←O		
city	string	City/Town.	→O ←O		
stateProvince	string	State or Province.	→O ←O		
postalCode	string	Postal Code.	→O ←O		
country	string	Country.	→M ←M		Enumeration = ISO Country Codes

5.6 Account Identifiers Object

In Mobile Money, there is no single and common method for identifying mobile money accounts and/or transaction parties. Identifiers include MSISDN (Mobile Number), Bank Sort Code, Account Number and Wallet Identifier. 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 Properties					
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		

5.7 Quotes Object

The quotes object defines the properties associated with domestic international remittance quotes.

Quotes Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
quoteId	string	The unique ID for this quote.	→NA ←M		
quoteExpiryTime	date-time	The timestamp when the quote will expire.	→NA ←O		
receivingServiceProvider	string	The name of the receiving service provider, i.e. the provider that the quote is associated with.	→NA ←O		
sendingAmount	string	Requested quotation amount as supplied by the sender.	→NA ←M		Please refer to Amount Validation
sendingCurrency	string	Currency of the requested quotation amount.	→NA ←M		Enumeration = ISO Currency Codes
receivingAmount	string	The total amount as it will be received by the receiving end user.	→NA ←M		Please refer to Amount Validation
receivingCurrency	string	The currency of the quote.	→NA ←M		Enumeration = ISO Currency Codes
fxRate	string	The conversion rate applicable between the sending and the receiving currency for the requested transaction.	→NA ←O		As per Amount Validation with the exception that up-to ten decimal places can be supplied.

deliveryMethod	string	The delivery method that is applicable to the quotation.	→NA ←O		Enumeration = Delivery Method Object
fees	array	Returns all fees that are applicable to the quote	→NA ←O	Fees Object	

5.8 Metadata Object

The metadata object allows additional properties to be specified for the parent object in the form of key/value pairs. Additional properties should only be used where no suitable defined property match can be found. The number of key/value pairs is limited to 20.

Identifier Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
key	string	Identifies the type of additional property.	→M ←M		
value	string	Identifies the value of the additional property.	→M ←M		

5.9 Supplementary Bill References Object

This object enables additional payment references to be specified for a bill payment in the form of key/value pairs. Additional properties should only be used where no suitable defined property match can be found. The number of key/value pairs is limited to 20.

Identifier Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
paymentReferenceType	string	Identifies the type of the additional payment reference.	→M ←M		
paymentReferenceValue	string	Identifies the value of the additional payment reference.	→M ←M		

5.10 Transaction Types 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 Properties					
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		

5.11 Channel Types 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 Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
channelType	string	Identifies the Channel Type.	→M ←M		

5.12 Fees Object

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

Account Identifier Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
feeType	string	Defines the type of fee.	→M ←M		
feeCurrency	string	Defines the currency for the given fee.	→M ←M		Enumeration = ISO Currency Codes
feeAmount	string	Defines the amount of the fee.	→M ←M		Please refer to Amount Validation

6 Enumerations

6.1 ISO Currency Codes

The three-character alphabetic code for currency as defined by ISO 4217 is to be used for all currency properties. 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.

6.2 Transaction Types

A small number of types have been defined to classify the nature of a transaction. Use of these types will enable clients to indicate the type of transaction in a manner that is common regardless of the API provider.

Code	Description
billpay	Payment of bill from a business for goods and/or services.
deposit	Exchange of cash in return for e-Money at a physical agent or via ATM
disbursement	Disbursement of funds (making payments from an organisation (business, NGO, government entity) to a mobile money recipient.
transfer	Transfer of funds between mobile money provider and another provider or financial institution in the same country.
merchantpay	Purchases of goods and/or services from shops (payer present) or online (payer not present).
inttransfer	Transfer of funds to a recipient in another country, either directly to/from a mobile wallet or via an international money transfer provider.
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.
withdrawal	Exchange of e-Money in return for cash at a physical agent or via ATM

6.3 ID Types

The ID Types enumeration contains accepted identification types. Due to the wide international variation in accepted types of identification, a catch-all type of ‘otherid’ is included.

ID Type	Description
passport	Payment of bill from a business for goods and/or services.
nationalregistration	National Registration Number.
othered	Catch-all for IDs not on the list.
drivinglicence	Driving Licence Number.
socialsecurity	Social Security Number.
alienregistration	Alien Registration ID.

nationalidcard	National Identity Card.
employer	Employers Identification.
taxid	Tax Identification Number.
seniorcitizenscard	Senior Citizens ID Card.
marriagecertificate	Marriage Certificate.
birthcertificate	Birth Certificate.
healthcard	Health Card.
votersid	Voters Identification.
villageelderletter	Letter of confirmation from village elder.
pancard	Credit/debit card number (Primary Account Number).
officialletter	Official letter confirming identity.

6.4 Account Identifiers

The Account Identifier enumeration lists all possible means to identify a target account and , the debit and/or credit party. 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.

6.5 ISO Country Codes

The two-character alphabetic code for country as defined by ISO 3166 is to be used for all properties specifying a country or nationality. The full list of codes is maintained by the International Organisation for Standardisation. The list can be obtained via the following website - http://www.iso.org/iso/country_codes.

6.6 Delivery Method Type

When a customer requests an international transfer quotation they are able to specify their preferred method of delivery of the transfer to the recipient. Permitted delivery methods are provided below.

Delivery Method	Description
directtoaccount	The transfer is to be delivered into the account (wallet) of the recipient.
agent	The recipient can visit an agent and get the transferred funds.
personaldelivery	a supplementary service where an authorised person can deliver the funds, in hand, to the receiving end user.

6.7 Frequency Type

When requesting a debit mandate, the API client is able to specific the frequency in which the payment should be taken. Valid values are defined in the table below.

Frequency Type	Description
weekly	Payment will be taken weekly.
fortnight	Payment will be taken every two weeks.
monthspecificdate	Payment to be taken on a specific date every month.
twomonths	Payment to be taken every two months.
threemonths	Payment to be taken every three months.
fourmonths	Payment to be taken every four months.
sixmonths	Payment to be taken every six months.
yearly	Payment to be taken yearly.
lastdaymonth	Payment to be taken on the last calendar day of the month.
lastdaymonthworking	Payment to be taken on the last working day of the month according to working days as per the resident country of the account.
lastmonday	Payment to be taken on the last Monday of the month.
lasttuesday	Payment to be taken on the last Tuesday of the month.
lastwednesday	Payment to be taken on the last Wednesday of the month.
lastthursday	Payment to be taken on the last Thursday of the month.

lastfriday	Payment to be taken on the last Friday of the month.
lastsaturday	Payment to be taken on the last Saturday of the month.
lastsunday	Payment to be taken on the last Sunday of the month.
specificdaymonthly	Payment to be taken on a specific day of the month.

7 API Behaviour & Error Handling

The Mobile Money API manages API state and exception handling in a harmonised manner:

- **Harmonised API State Model**. For synchronous requests, HTTP response codes are used exclusively to manage request state communication – both during the creation of a resource and the update of a resource. For asynchronous requests, a RequestState object is additionally used to manage request state communication.
- **Harmonised HTTP Status Codes**. Standardised HTTP status codes are associated with a harmonised state model for resource creation and update.
- **Harmonised Application-level Error Reason Codes**. Standardised error codes are returned in a standardised error object for ALL client-responsible and server-responsible errors. Error parameters allow more detailed provider-specific diagnostic information to be returned with the error.
- **API Heartbeat** enables API provider service availability to be established by clients on a regular basis.
- The **responses** object enables clients to re-request a missing API response.
- **Error Handling Guidelines** provide a standardised way of handling exceptions, including managing timeouts and retries.

7.1 Use of HTTP Response Codes

Response status is represented by the use of HTTP response codes. Figure 8.1 illustrates the valid response codes for synchronous requests. The response codes for asynchronous requests are represented by the sequence flows in the API Behavioural Model.

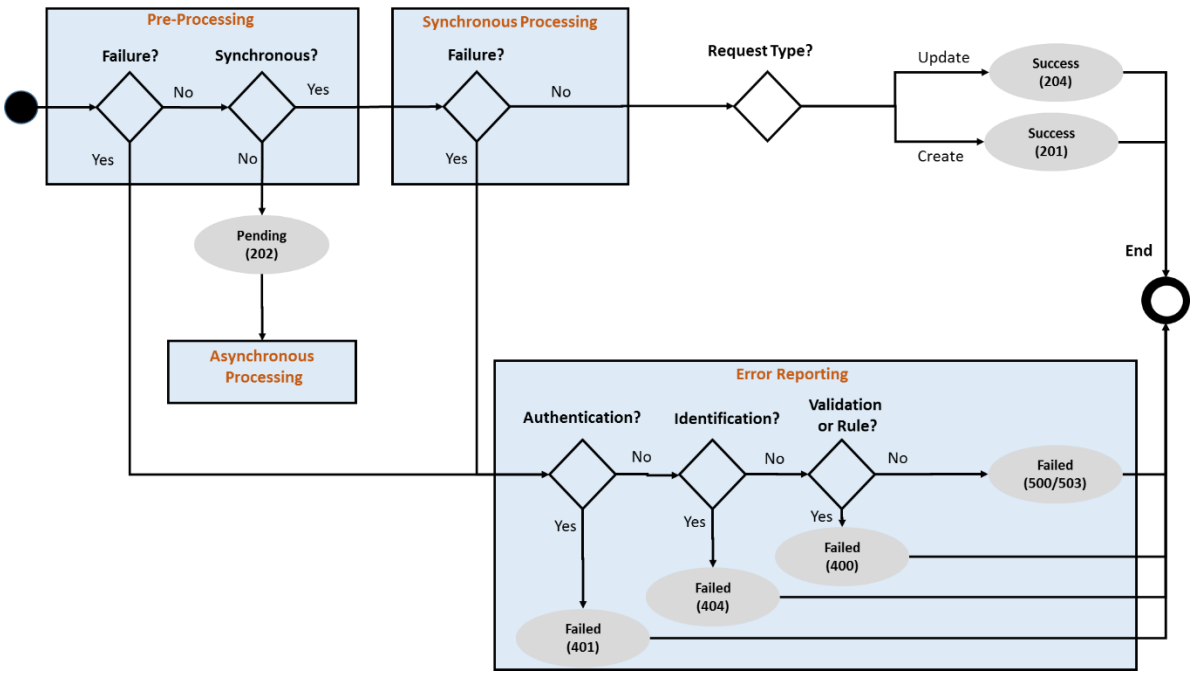

Figure 2: Use of HTTP Response Codes for Synchronous Processing

7.2 API Behavioural Model

7.2.1 Overview

API behaviour is governed by the following factors:

- The resource.
- The type of operation, i.e. create, update or read.
- Whether the provider will process the request synchronously.
- Whether the provider implements callback or polling methods for asynchronous processing.

When combined, these factors drive API behaviour from the simplest request (e.g. view an account balance) to a more complex request (e.g. update a debit mandate asynchronously).

7.2.2 Request State Object

Asynchronous flows involve a callback or a polling mechanism to enable the client to determine the final state of the request. Both mechanisms involve the use of the RequestState object as per below:

- **Callback**. A request is initiated via a HTTP POST (create resource) or PATCH (update resource) request with an intermediate response represented by a RequestState object. Once the request has been completed, the provider will initiate a PUT request to the URL specified by the client in the X-Callback-URL request header. The callback will provide the client with one of the following:
    - Final representation of the resource for successful creation requests
    - A simple confirmation of update for successful update requests
- **Polling**. Is initiated by a HTTP POST or PATCH request with an intermediate response provided in the form of the RequestState object. A HTTP GET is then issued on the RequestState by the client at regular defined intervals until the final resource state and resource reference is returned.

The object definition for RequestState is described below.

RequestState Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
serverCorrelationId	string	A unique identifier issued by the provider to enable the client to identify the RequestState resource on subsequent polling requests.	→NA ←M		UUID format
status	string	Indicates the status of the request.	→NA ←M		Enumeration = pending, completed, failed

pendingReason	string	A textual description that can be provided to describe the reason for a pending status.	→NA ←O		
notificationMethod	date-time	Indicates whether a callback will be issued or whether the client will need to poll.	→NA ←M		Enumeration = callback, polling
objectReference	string	Provides a reference to the subject resource, e.g. transaction reference.	→NA ←O		
expiryTime	date-time	Indicates the time by which the provider will fail the request if completion criteria have not been met. For an example, a debit party failing to authorise within the allowed time period.	→NA ←O		
pollLimit	integer	Indicates the number of poll attempts for the given requeststate resource that will be allowed by the provider.	→NA ←O		
errorReference	object	If the asynchronous processing failed, details of the error will be returned here.	→NA ←O	Errors Object	

7.2.3 API Sequence Flow Patterns

There are seven harmonised Mobile Money API sequence flows. These are reflected figure 3 and show the necessary HTTP methods, request object, response object and expected success and failure response codes for each API service.

				Leg 1							Leg 2							Leg 3						
							Success Response		Failure Response					Success Response		Failure Response					Success Response		Failure Response	
Endpoint	Operation	Type	Notification Method	HTTP Operation	Invoked By	Request Object	Response Object	HTTP Response Code	Response Object	HTTP Response Code	HTTP Operation	Invoked By	Request Object	Response Object	HTTP Response Code	Response Object	HTTP Response Code	HTTP Operation	Invoked By	Request Object	Response Object	HTTP Response Code	Response Object	HTTP Response Code
/Resource	Read	Sync	N/A	GET	Client	N/A	As per resource	200	Error	4xx, 5xx	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
/Resource	Create	Sync	N/A	POST	Client	As per resource	As per resource	201	Error	4xx, 5xx	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
/Resource	Create	Async	Callback	POST	Client	As per resource	RequestState	202	Error	4xx, 5xx	PUT	Provider	As per resource for completed requests OR	None	204	Error	4xx, 5xx	N/A	N/A	N/A	N/A	N/A	N/A	N/A
/Resource	Create	Async	Poll	POST	Client	As per resource	RequestState	202	Error	4xx, 5xx	GET	Client	N/A	RequestState	200*	Error	4xx, 5xx	GET	Client	N/A	As per Resource	200	Error	4xx, 5xx
/Resource	Update	Sync	N/A	PATCH	Client	Generic PATCH	None	204	Error	4xx, 5xx	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
/Resource	Update	Async	Callback	PATCH	Client	Generic PATCH	RequestState	202	Error	4xx, 5xx	PUT	Provider	Simple Result Message	None	204	Error	4xx, 5xx	N/A	N/A	N/A	N/A	N/A	N/A	N/A
/Resource	Update	Async	Poll	PATCH	Client	Generic PATCH	RequestState	202	Error	4xx, 5xx	GET	Client	N/A	RequestState	200*	Error	4xx, 5xx	GET	Client	N/A	As per Resource	200	Error	4xx, 5xx

Figure 3: Supported Mobile Money API Sequence Flow Patterns Matrix

7.2.4 API Sequence Flow Diagrams

Figures 4 to 10 illustrate the standard flow patterns for the Mobile Money API. The green flows represent a success path and red flows represent a failure path. A solid line indicates a HTTP request and a dotted line indicates a HTTP response.

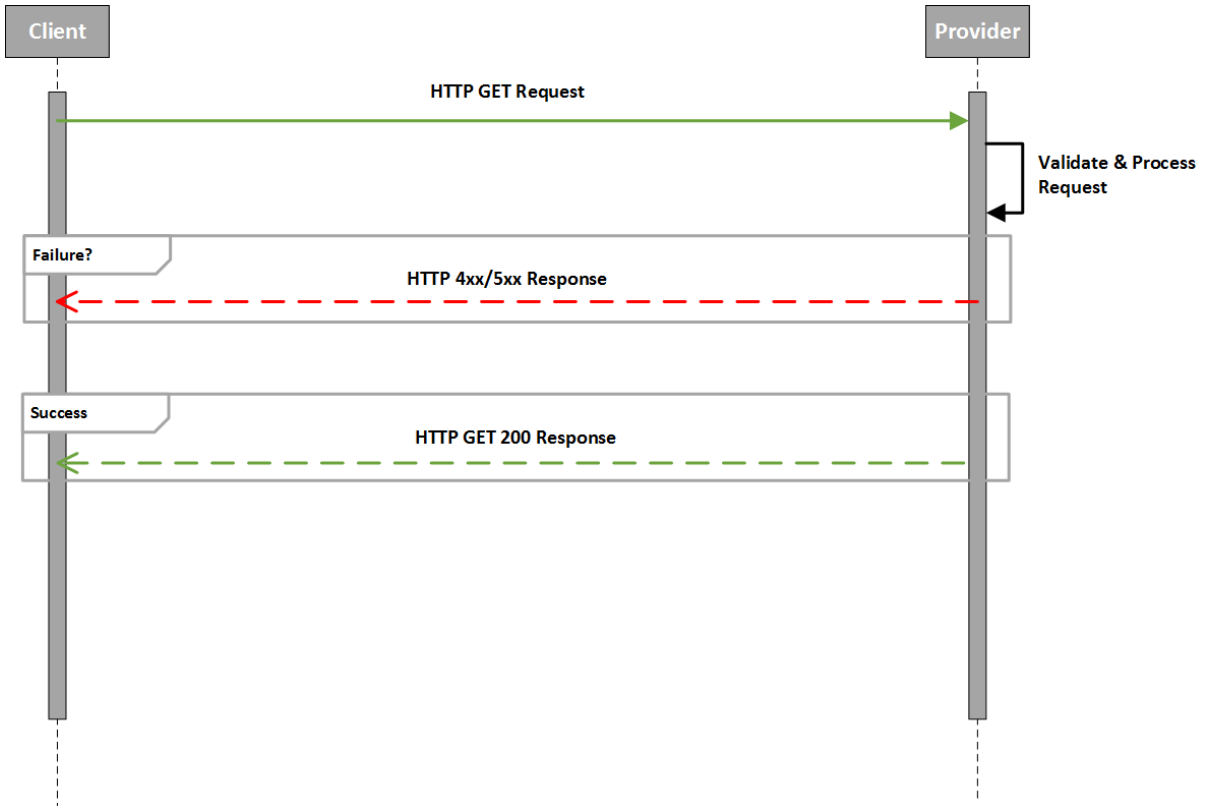

Figure 4: Read (GET) Flow Pattern

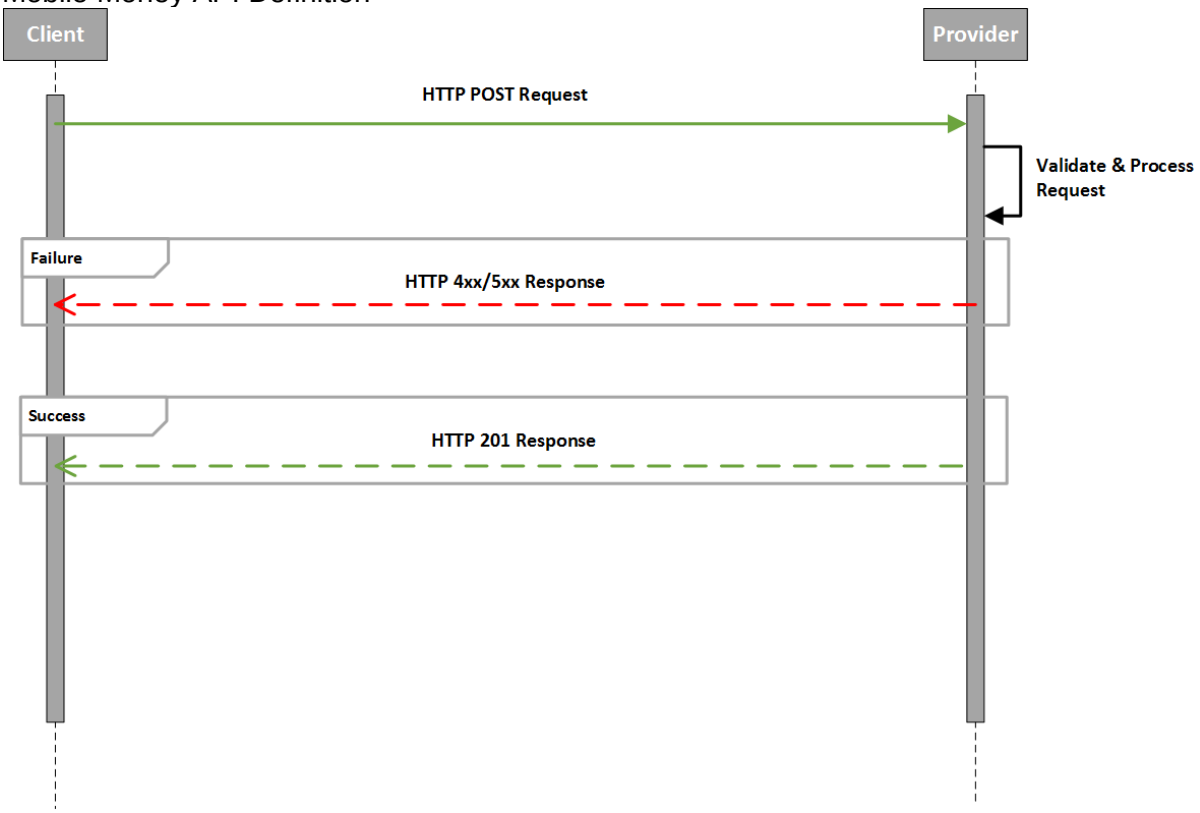

Figure 5: Create (POST) Synchronous Flow Pattern

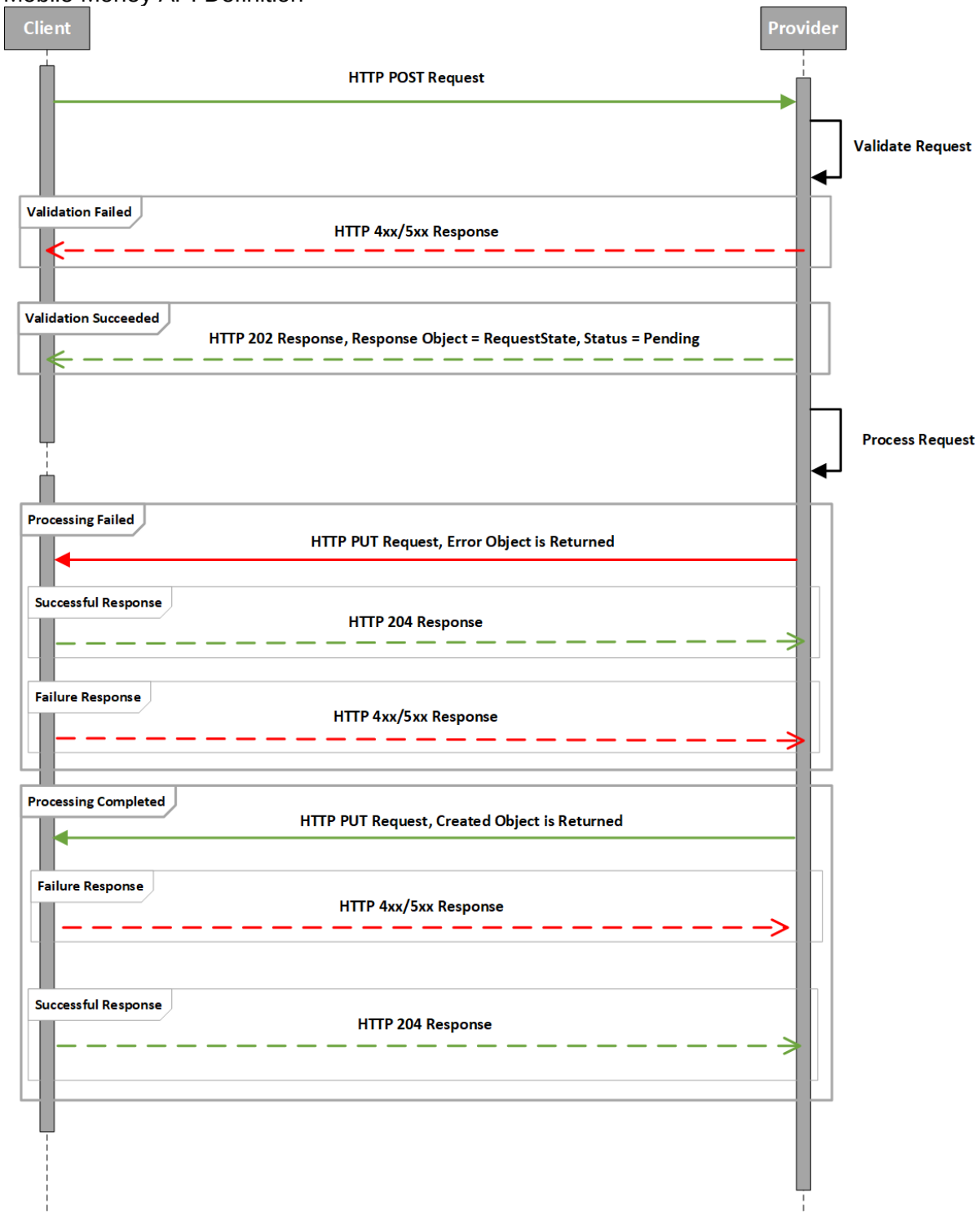

Figure 6: Create (POST) Callback Asynchronous Flow Pattern

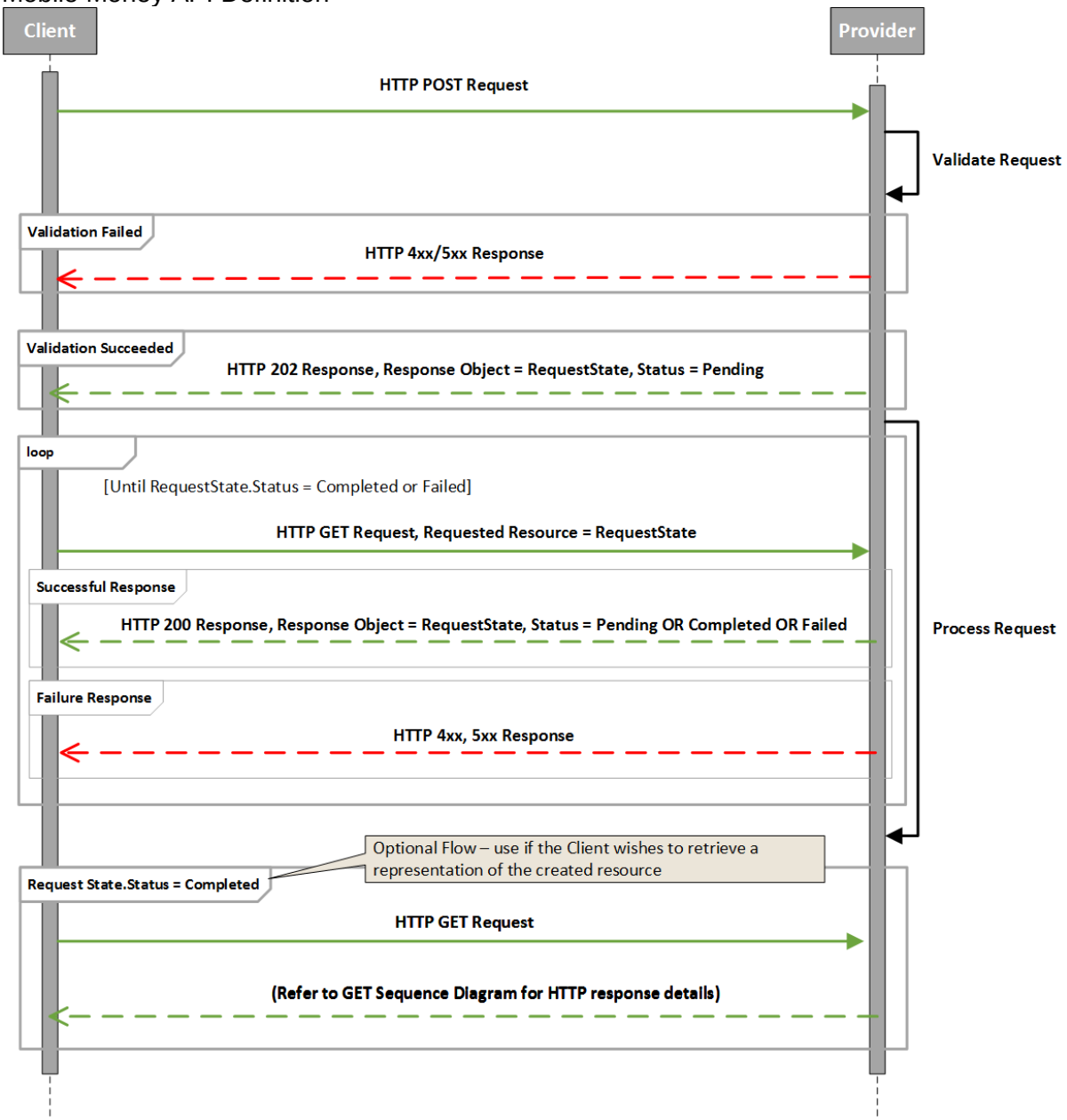

Figure 7: Create (POST) Polling Asynchronous Flow Pattern

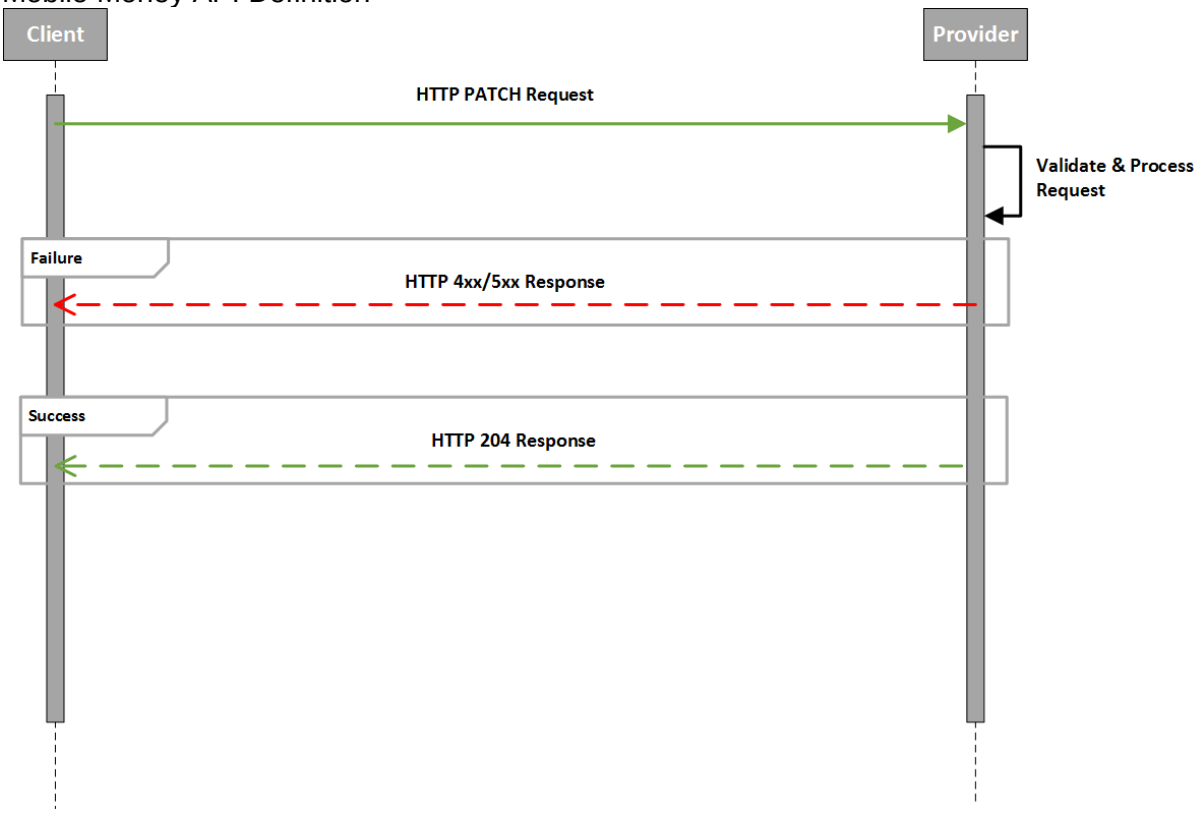

Figure 8: Update (PATCH) Synchronous Flow Pattern

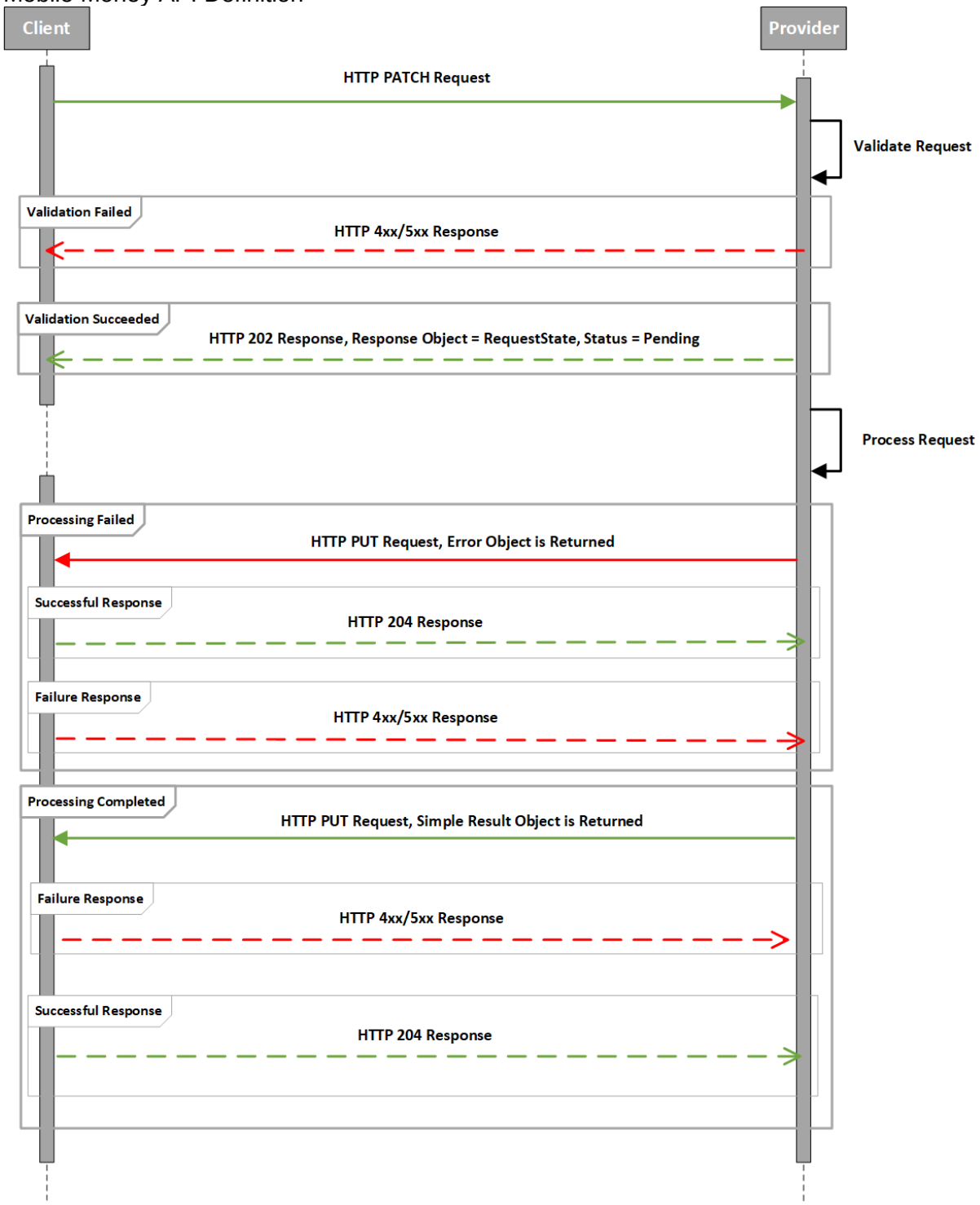

Figure 9: Update (PATCH) Callback Asynchronous Flow Pattern

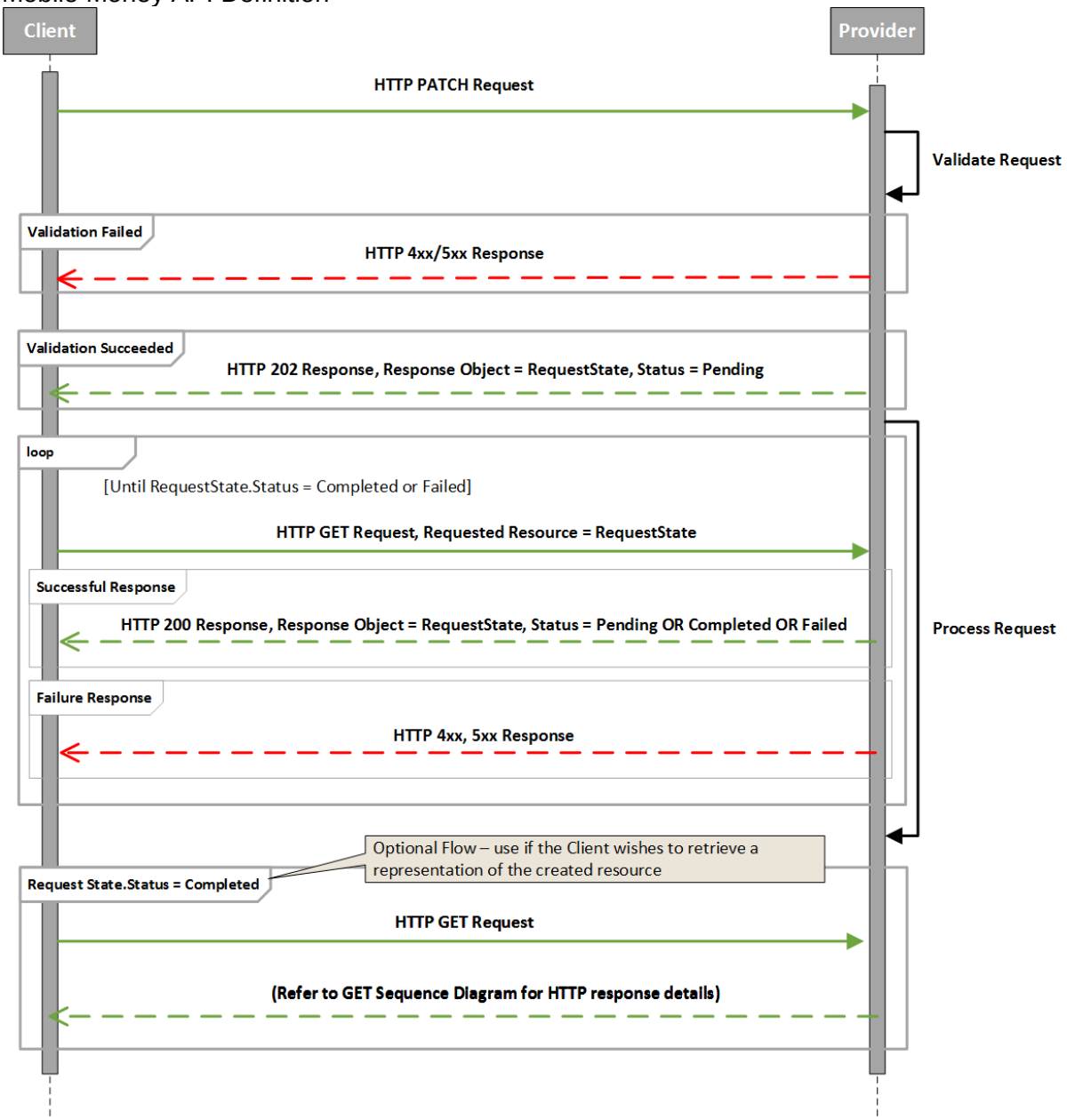

Figure 10: Update (PATCH) Polling Asynchronous Flow Pattern

7.3 HTTP Status Responses and Error Categories

The following HTTP status codes are returned for the listed methods:

Method	Success	Intermediate*	Client Error	Server Error
GET	200	N/A	400, 401, 404	500, 503
PATCH	204	202	400, 401, 404	500, 503
POST	201	202	400, 401, 404	500, 503

*Note that 'intermediate' column in the table above relates to the HTTP response for the first leg of an asynchronous request.

7.3.1 Client Error Categories

Client errors fall into one of four categories as shown below:

Error Category	Description	HTTP Response Code
BusinessRule	The resource could not be successfully completed due a violation of a business rule. Business rules include financial limit violations, duplicate requests and invalid states.	400
Validation	Violation of a constraint that will prevent the resource from being processed. Examples include: - Invalid property length. - Failure to resolve regular expression.	400
Authorisation	It was not possible to authenticate or authorise the client or other party to perform the service.	401
Identification	The requested resource could not be matched on the system with the supplied identifier(s).	404

Additional non-harmonised 4xx HTTP response codes may be returned to the client by web or proxy servers.

Where the need for prevention of information leakage is paramount and this outweighs the benefits of error granularity, the provider may wish to use a 400 response code with no error body.

7.3.2 Server Error Categories

Server Errors are captured in two simple categories as per the following table:

Error Category	Description	HTTP Response Code
Internal	The request could not be completed due to non-client related issues that do not constitute complete system unavailability. Examples include software licence issues, unavailability of system configuration information.	500

Service Unavailable	The service is not currently available. This could be due to network issues, issues with individual components or complete systems outages. Regardless of the cause, the result means that the request cannot be performed.	503

Additional non-harmonised 5xx HTTP response codes may be returned to the client by web or proxy servers.

7.4 Error Codes Definition

The mobile money API uses harmonised error codes in order to provide consistent error reporting back to API clients. Error codes are structured as per follows:

- Error Category. All error codes must be associated with a client or server error category. This provides context and uniqueness to the error code.
- Error Code. Provides the reason for the request failure. A human-readable description of the error is also available. Error codes are generally granular, i.e. they provide a specific reason for failure. In some cases, granularity is neither possible or desirable. Where this is the case, a 'Generic' code has been defined. Cases include:
    - The API Provider has generated an error that does not map to existing Rule error codes.
    - The API Provider wishes to avoid disclosure of confidential information regarding the resource or parties to the resource. For example, the fact that a customer has breached their monthly transaction limit may not be disclosed to specific clients.
- Error Parameters. Provides a construct to communicate supplementary information regarding the error in key/value pairs. The supplementary information is currently non-harmonised and can include:
    - The API provider-specific Error Code and Description.
    - Additional identification of the error subject, e.g. account identifiers, invalid properties etc…
    - Diagnostic information, e.g. affected subsystem, licence failure type etc…

Although error parameter contents are non-harmonised, care should be taken regarding confidentially of disclosed information. Confidential parameter information should only be disclosed to trusted clients.

A full list of harmonised Error Codes is provided here.

7.5 Errors Object Definition

All errors generated by the API provider will contain an Errors object in the response body. At a minimum, the Errors object must contain the Error Reason and Error Code.

Errors Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
errorCategory	string	The category grouping for the error.	→M ←M		Enumeration = Error Categories
errorCode	string	The harmonised error code identifying the reason for error.	→M ←M		Enumeration = Error Reasons
errorDescription	string	A textual description of the error.	→O ←O		
errorDateTime	date-time	The timestamp indicating when the error occurred.	→O ←O		
errorParameters	array	Diagnostic information in the form of key/value pairs relating to the error.	→O ←O	Metadata	

7.6 API Heartbeat

The Heartbeat API is used for monitoring purposes and establishes whether the system of an API provider is in a state that enables a client to submit a request for processing within established SLAs. There are three states that can be returned by the API provider in response to a heartbeat request:

- Available. The system is available and can receive and complete requests within SLAs.
- Degraded. The system can receive and complete requests but not within SLAs, i.e. delay in transaction processing is anticipated. When known, the expected processing delay time can be returned by the provider.
- Unavailable. The system cannot receive and process requests. Any submitted requests will fail whilst the system is in this state.

The Heartbeat can be requested using the following method and path:

- ***GET /heartbeat***

Only synchronous API Heartbeat requests are supported. The HTTP response contains the following properties.

Heartbeat Response Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
serviceStatus	string	Provides the status of the requested service.	→NA ←M		Enumeration = available, unavailable, degraded
delay	number	The anticipated processing delay in milliseconds.	→NA ←O		serviceStatus property must be set to degraded.
plannedRestorationTime	date-time	Where the planned restoration time is known (e.g. scheduled maintenance), it can be provided in this property.	→NA ←O		

7.7 Missing Response Retrieval

In some circumstances, the client may not have received the final representation of the resource for which it attempted to create or update. For example, a proxy server issue may have resulted in a HTTP 5xx response but the provider may have actually successfully completed the request. The ***/responses*** API allows a client to identify and retrieve the final representation of the resource assuming that the resource was created. In order to get a representation, the client issues a ***GET /Responses/{clientCorrelationId}***. The provider will then match the client correlation ID to the appropriate resource and return a link to that resource. If the resource is not found for the given correlation ID then a HTTP 404 will be returned.

The response object for ***/responses*** is detailed below.

Responses Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
link	string	Provides a URL to the resource associated with the given correlation ID.	→NA ←M		

7.8 Harmonised Error Codes

The error codes defined in this section are considered a basic initial set and will be expanded over time.

Error Category	Error Code	Error Code Description
BusinessRule	GenericError	A generic Error Code for the Rule Error Category. This is used in two scenarios: 1. The API Provider has generated an error that does not map to existing Rule error codes. 2. The API Provider wishes to avoid disclosure of confidential information regarding the resource or parties to the resource.
BusinessRule	DailyVolumeLimitExceeded	The party has exceeded their daily transacting volume limit - This can be a service limit or a limit that is specific to the party.
BusinessRule	DailyValueLimitExceeded	The party has exceeded their daily transacting value limit - This can be a service limit or a limit that is specific to the party.
BusinessRule	WeeklyVolumeLimitExceeded	The party has exceeded their weekly transacting volume limit - This can be a service limit or a limit that is specific to the party.
BusinessRule	WeeklyValueLimitExceeded	The party has exceeded their weekly transacting value limit - This can be a service limit or a limit that is specific to the party.
BusinessRule	MonthlyVolumeLimitExceeded	The party has exceeded their monthly transacting volume limit - This can be a service limit or a limit that is specific to the party.
BusinessRule	MonthlyValueLimitExceeded	The party has exceeded their monthly transacting value limit - This can be a service limit or a limit that is specific to the party.
BusinessRule	AccountMaxTotalValueExceeded	The party has exceeded their cumulative transacting value limit defined for the account.
BusinessRule	AccountMaxTotalVolumeExceeded	The party has exceeded their cumulative transacting volume limit defined for the account.
BusinessRule	LessThanTransactionMinValue	The amount specified for the transaction is less than the defined minimum for the service.
BusinessRule	GreaterThanTransactionMaxValue	The amount specified for the transaction is greater than the defined maximum for the service.
BusinessRule	MaxBalanceExceeded	The amount specified will cause the balance of the credit parties account to exceed the rule limit.
BusinessRule	SamePartiesError	The debit and credit parties are the same.
BusinessRule	DuplicateRequest	The request has previously been processed, i.e. this request is a duplicate and hence has been rejected.

BusinessRule	InsufficientFunds	Available funds are not sufficient to enable the party to be debited for the requested transaction.
BusinessRule	IncorrectState	The account is in a state that does not permit the requested service.
BusinessRule	UnderPaymentNotAllowed	The requested amount is less than the amount that needs to be supplied for this transaction
BusinessRule	OverPaymentNotAllowed	The requested amount is greater than the amount that needs to be supplied for this transaction
BusinessRule	RateLimitError	The client has submitted too many requests within a period of time.
BusinessRule	TransactionTypeError	The harmonised transaction type is not supported by the API Provider for the given request.
BusinessRule	NoMandateAuthority	The transaction request is not supported by the debit mandate.
BusinessRule	LinkViolation	The request has violated an account to account link for one of the following reasons: - Pull request not supported by the Link Mode. - Push request not supported by the Link Mode.
BusinessRule	CountryofOriginNotPermitted	The API Provider does not allow transactions to be accepted from the specified country of origin.
BusinessRule	NationalityNotPermitted	The API Provider does not allow transactions to be accepted from individuals with the specified nationality.
BusinessRule	IdDocumentNotSupported	The API Provider does not allow the specified ID Document.
BusinessRule	IssuingCountryNotSupported	The API Provider does not allow the issuing country of the specified ID document.
QuoteExpiry	QuoteHasExpired	A transaction with a quote reference cannot be processed at the quote has expired.
Identification	IdentifierError	The requested resource could not be matched on the system with the supplied identifier(s).
Validation	GenericError	A generic Error Code for the Validation Error Category. This is used in where the API Provider has generated an error that does not map to existing Validation error codes.
Validation	LengthError	The specified property contents are greater than the maximum allowed length or less than the minimum allowed length.
Validation	FormatError	The specified property contents do not conform to the format required for this Property.
Validation	NegativeValue	The amount supplied is negative and this is not allowed for the given service.
Validation	CurrencyNotSupported	The currency supplied is not supported by the API Provider.
Validation	MandatoryValueNotSupplied	A mandatory value has not been provided in the header and/or JSON body.

Validation	InvalidOffset	The pagination offset provided is not valid. This is typically because the offset is greater than the maximum number of records.
Authorisation	ClientAuthorisationError	General Client Authentication failure. No further details provided to prevent leakage of security information.
Authorisation	RequestDeclined	The debit party did not approve the request.
Authorisation	ServicingPartyAuthorisationError	The servicing party (e.g. agent) credentials or permissions are not valid.
Authorisation	RequestingPartyAuthorisationError	The party requesting the service has not provided the right credentials and/or does not have permission to perform this service.
Internal	GenericError	The request could not be completed due to a non-client related issues that do not constitute complete system unavailability. Examples include software licence issues, unavailability of system configuration information.
Service Unavailable	GenericError	The service is not currently available. This could be due to network issues, issues with individual components or complete systems outages. Regardless of the cause, the result means that the request cannot be performed.

This document is produced by the GSMA with input from the GSMA Mobile Money API Technical Steering Committee. 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.