Mobile Money API Specification 1.2.0 Fundamentals

Document Summary

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

Document History

Document Version	API Release Version	Date	Brief Description of Change	Editor / Company
0.1	1.2.0-beta	Aug 2020	• Initial draft of document	GSMA
0.2	1.2.0	Oct 2020	• Implemented RFC 2020-18	GSMA

Other Information

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

Table of Contents

1 Introduction

The purpose of this document is to specify the design principles, 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 will reduce integration complexity across the mobile money ecosystem.

1.1 Mobile Money API Design Principles

The Mobile Money API is based upon the following design principles:

- Use of REST principles.
- Provide a set of harmonized objects that are abstracted from the underlying object representations held in mobile money systems. This allows an API client to construct an API message without requiring specific knowledge of the target server implementation.
- Use of standardised enumerations, removing the need for developers to map for each and every API implementation. This includes use of ISO international standards for enumerators such as currency and country code.
- Use of supplementary key/value properties to enable use case and/or mobile money provider-specific properties to be conveyed where necessary.

This documentation contains the following sections:

- **API Fundamentals**. The core principles and constructs that underpin the API.
- **API Behaviour and Error Handling**. Describes behavioural aspects of the API and details error handling mechanisms including error code definition, Heartbeat API definition and a detailed explanation of synchronous and asynchronous methods.

For further reading, please refer to the following documents:

- **Mobile Money API Introduction**. Introduces the use and benefits of the Mobile Money API. Also provides a glossary of terms used by the Mobile Money API specifications.

- **Mobile Money API Specification**. Documents the Mobile Money API endpoints, fields, objects, and enumerations.

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

2 API Fundamentals

2.1 URI

The Mobile Money API uses 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.

2.2 Methods

The specification supports the following request methods:

- **POST**. Used to create a resource
- **PATCH**. Used to update a resource
- **GET**. Used to return a representation of a resource or a collection of resources.

2.3 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 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.

2.4 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, for example, */accounts/bills*.

2.5 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 UUID that enables the client to correlate the API request with the resource created/updated by the provider. The client correlation id is specified in the HTTP Request Header. When a provider issues a callback, the provider should ensure that the original correlation id provided by the client is placed in request header.

The client 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.

2.6 Use Case Flow Patterns

All use cases supported by the API are built on standardised flow patterns. Flow patterns exist for viewing, updating, and creating resources.

There are two standard flows:

- **Synchronous Flow**. The final resource is always provided in response to an API request. There is no interim response. Can be used with POST, PATCH and GET requests.
- **Asynchronous Flow**. An interim response is always provided in response to an API request in the form of a Request State object. The final response is then provided via a callback or alternatively can be accessed via polling on Request State. Can be used with POST and PATCH requests.

More detail can be found in the API Behavioural Model section.

2.7 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.

2.8 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.

2.8.1 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	

2.8.2 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.

2.8.3 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	Conditional	Used to identify the API invocation channel.

	example, USSD, Web, App..		
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.

2.8.4 Custom Response Headers

Header	Value	Optionality	Notes
X-Date	{The date and time that the response was sent in HTTP-date format - RFC 7231}	Conditional	Used for Basic message integrity checks.

2.9 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

2.10 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 999999999999999999.9999.

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
000001.32	No

3 API Behaviour & Error Handling

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

- **API Behavioural Model**. Provides standardised flows for synchronous and asynchronous processing patterns.
- **API Error Handling**. Supports use of standardised HTTP response codes and an ***errors*** object that returns additional information as to why the request failed.
- **API Heartbeat** enables API provider service availability to be established by clients.
- The ***responses*** object enables clients to request a missing API response.

3.1 API Behavioural Model

3.1.1 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.

3.1.2 Request State Object

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

- **Callback**. A request is initiated via a HTTP POST or PATCH request with an intermediate response represented by a Request State 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 *{"result": "success"}* response for successful update requests

- **Polling**. A request is initiated via a HTTP POST or PATCH request with an intermediate response provided in the form of the Request State object. A HTTP GET is then issued against */requeststates/{serverCorrelationId}* endpoint by the client at 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
objectReference	string	Provides a reference to the subject resource, e.g. transaction reference.	→NA ←O		
status	string	Indicates the status of the request.	→NA ←M		Enumeration = pending, completed, failed
notificationMethod	date-time	Indicates whether a callback will be issued or whether the client will need to poll.	→NA ←M		Enumeration = callback, polling
pendingReason	string	A textual description that can be provided to describe the reason for a pending status.	→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	

3.1.3 Generic API Sequence Diagrams

Figures 1 to 7 illustrate the standard flows for the Mobile Money API. The green flows represent a success path and red flows represent a failure path.

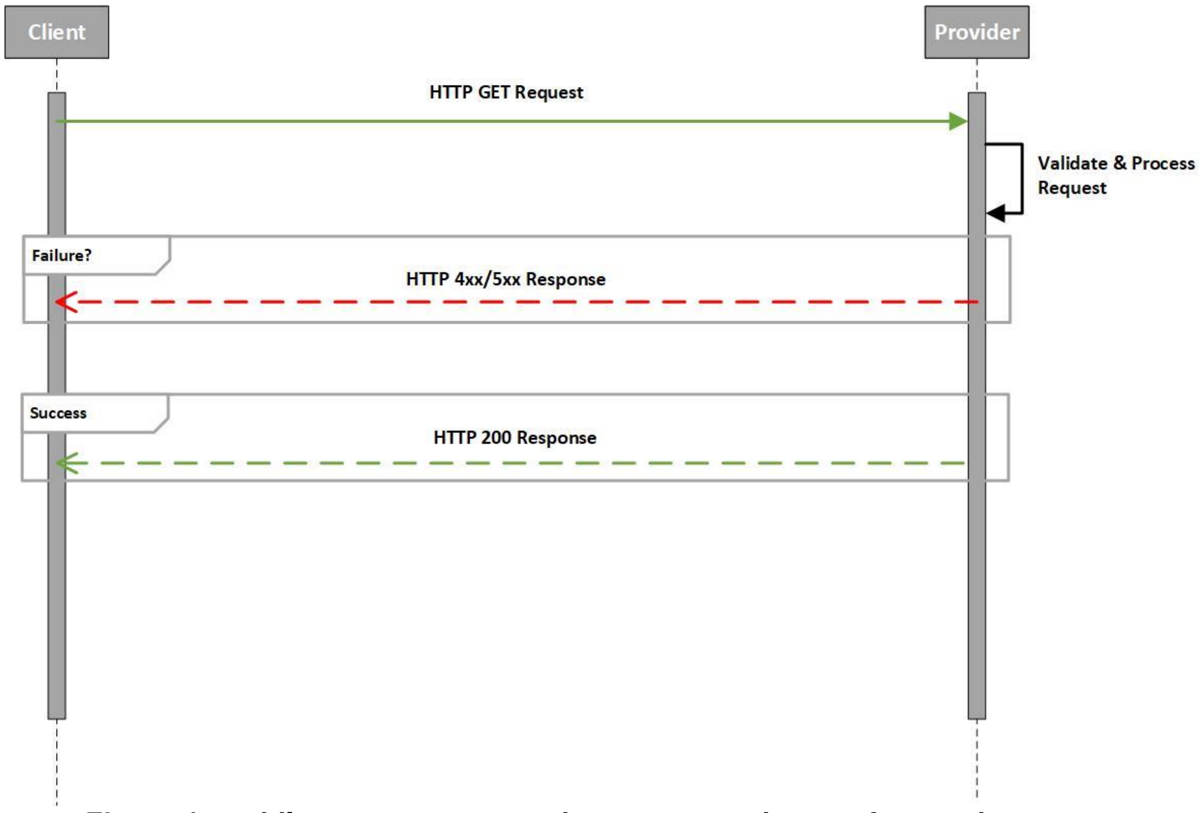

Figure 1: Mobile Money API Generic Sequence Diagram for Read Requests

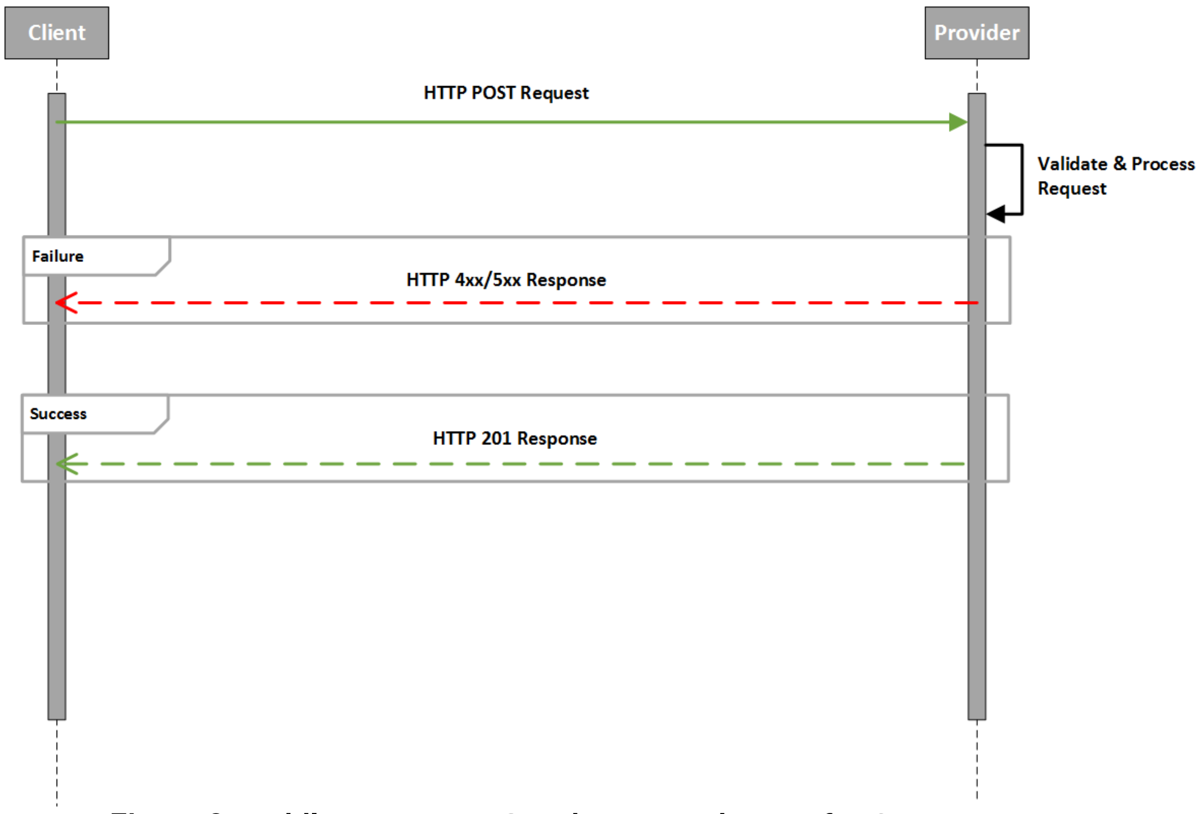

Figure 2: Mobile Money API Synchronous Diagram for Create Requests

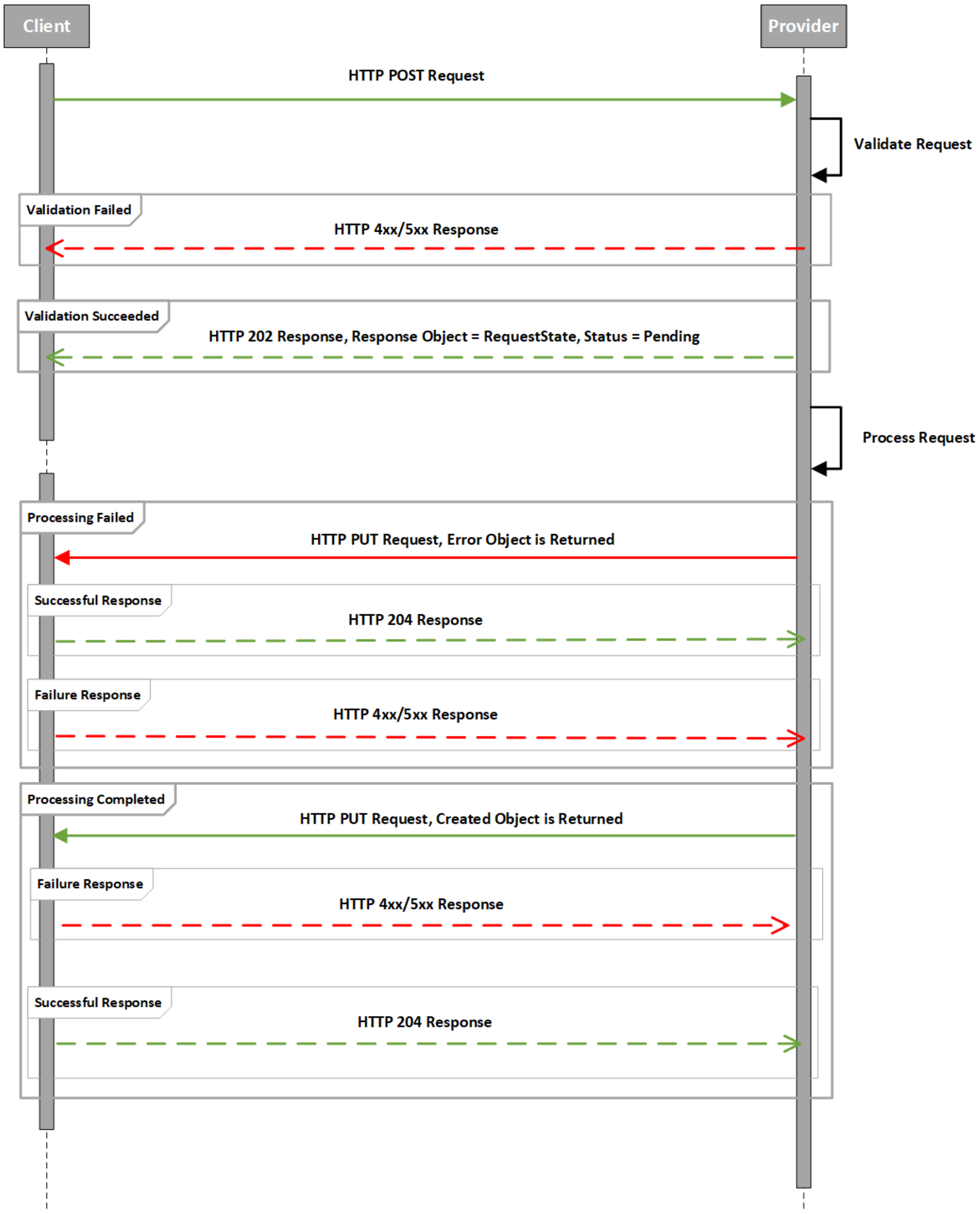

Figure 3: Mobile Money API Asynchronous Callback Diagram for Create Requests

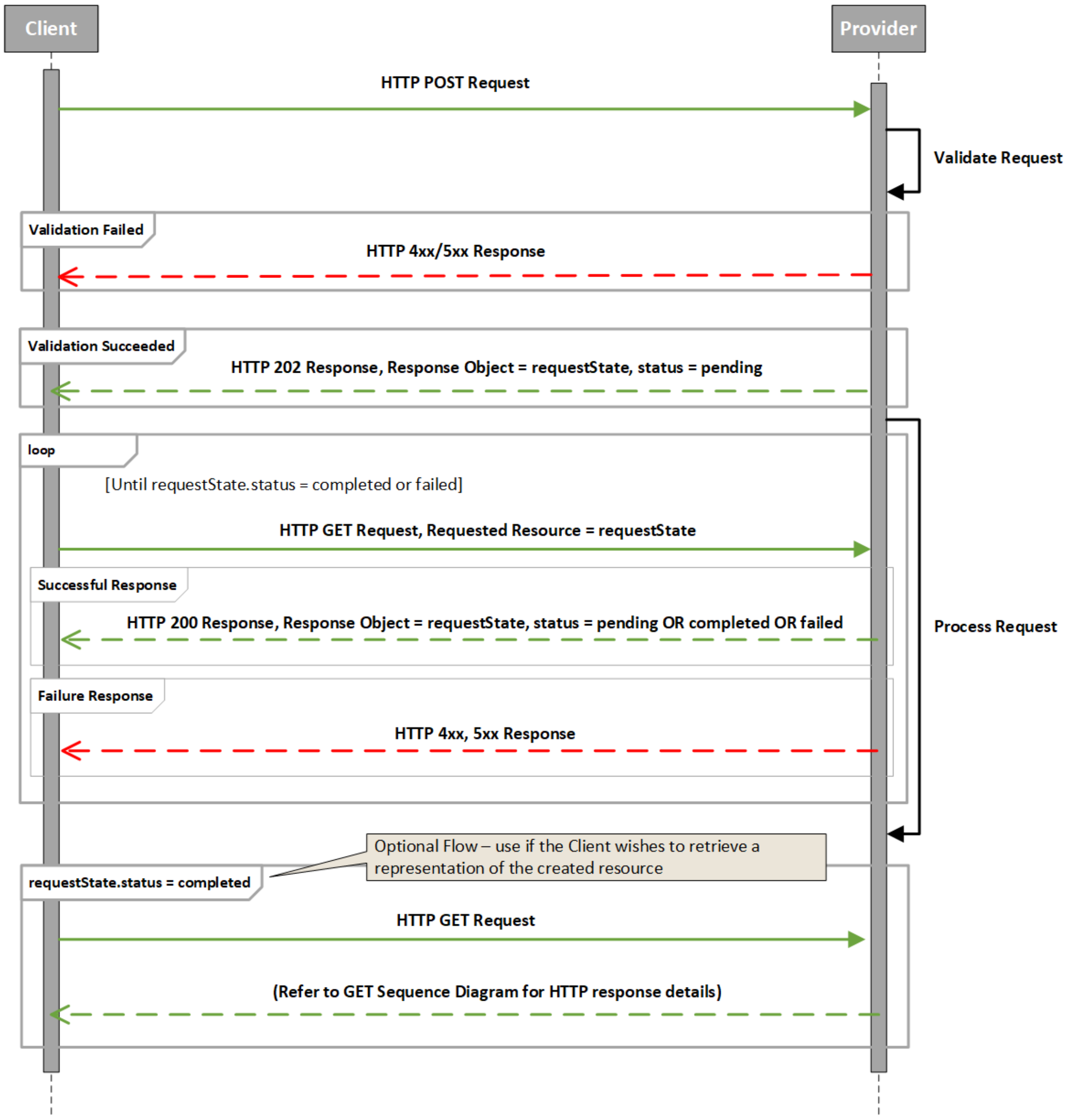
Client
Provider
HTTP POST Request
Validate Request
Validation Failed
HTTP 4xx/5xx Response
Validation Succeeded
HTTP 202 Response, Response Object = requestState, status = pending
loop
[Until requestState.status = completed or failed]
HTTP GET Request, Requested Resource = requestState
Successful Response
HTTP 200 Response, Response Object = requestState, status = pending OR completed OR failed
Failure Response
HTTP 4xx, 5xx Response
Process Request
Optional Flow – use if the Client wishes to retrieve a representation of the created resource
requestState.status = completed
HTTP GET Request
(Refer to GET Sequence Diagram for HTTP response details)

Figure 4: Mobile Money API Asynchronous Polling Diagram for Create Requests

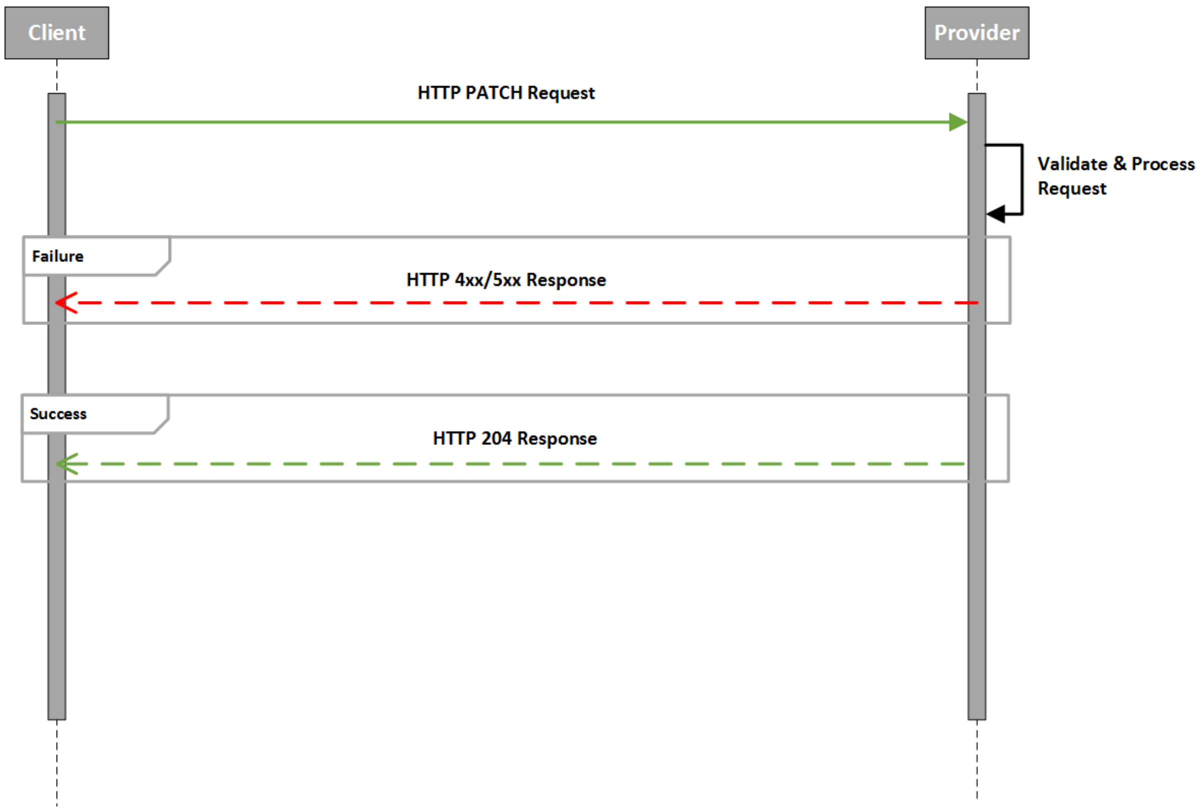

Figure 5: Mobile Money API Synchronous Diagram for Update Requests

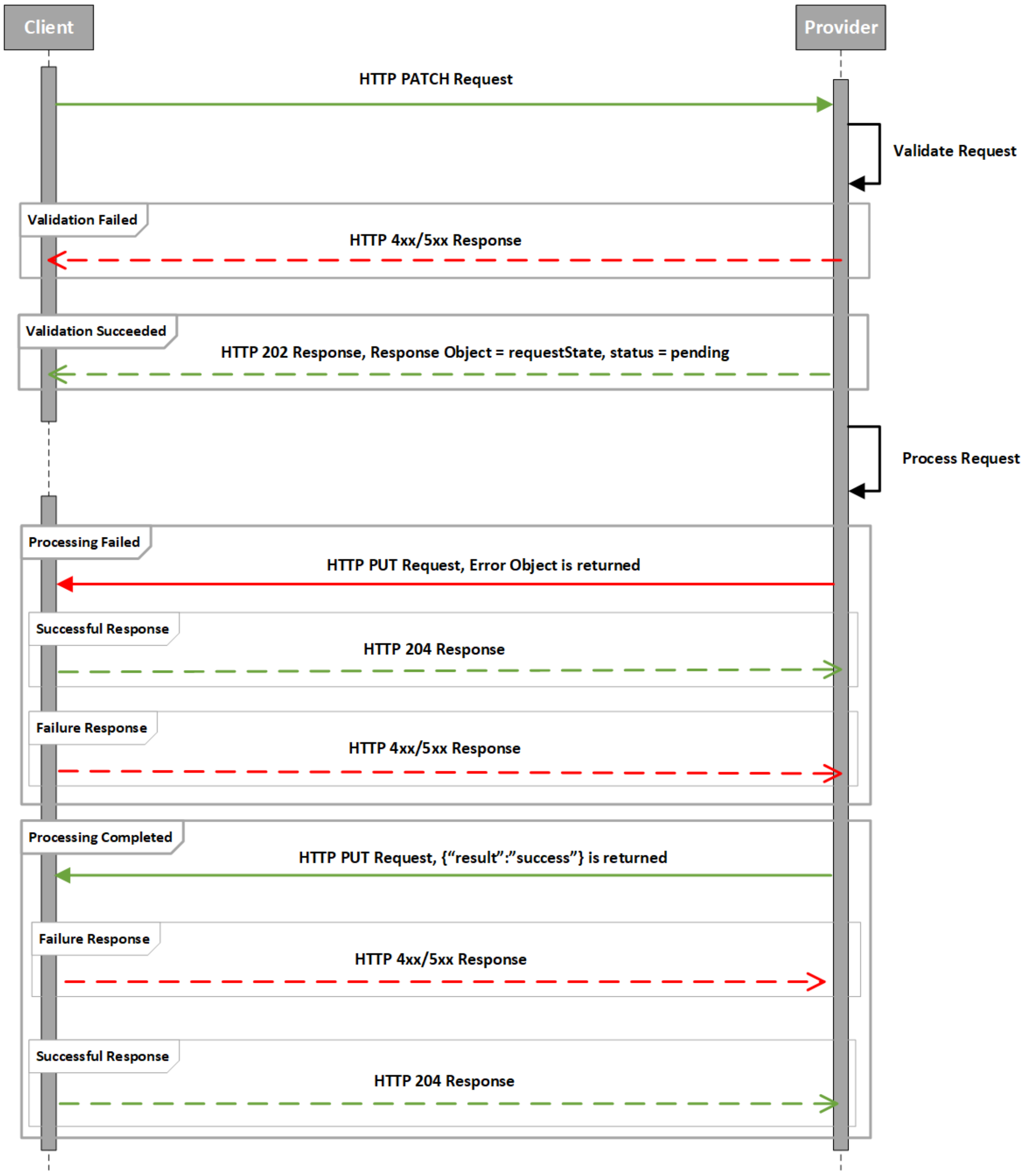

Figure 6: Mobile Money API Asynchronous Callback Diagram for Update Requests

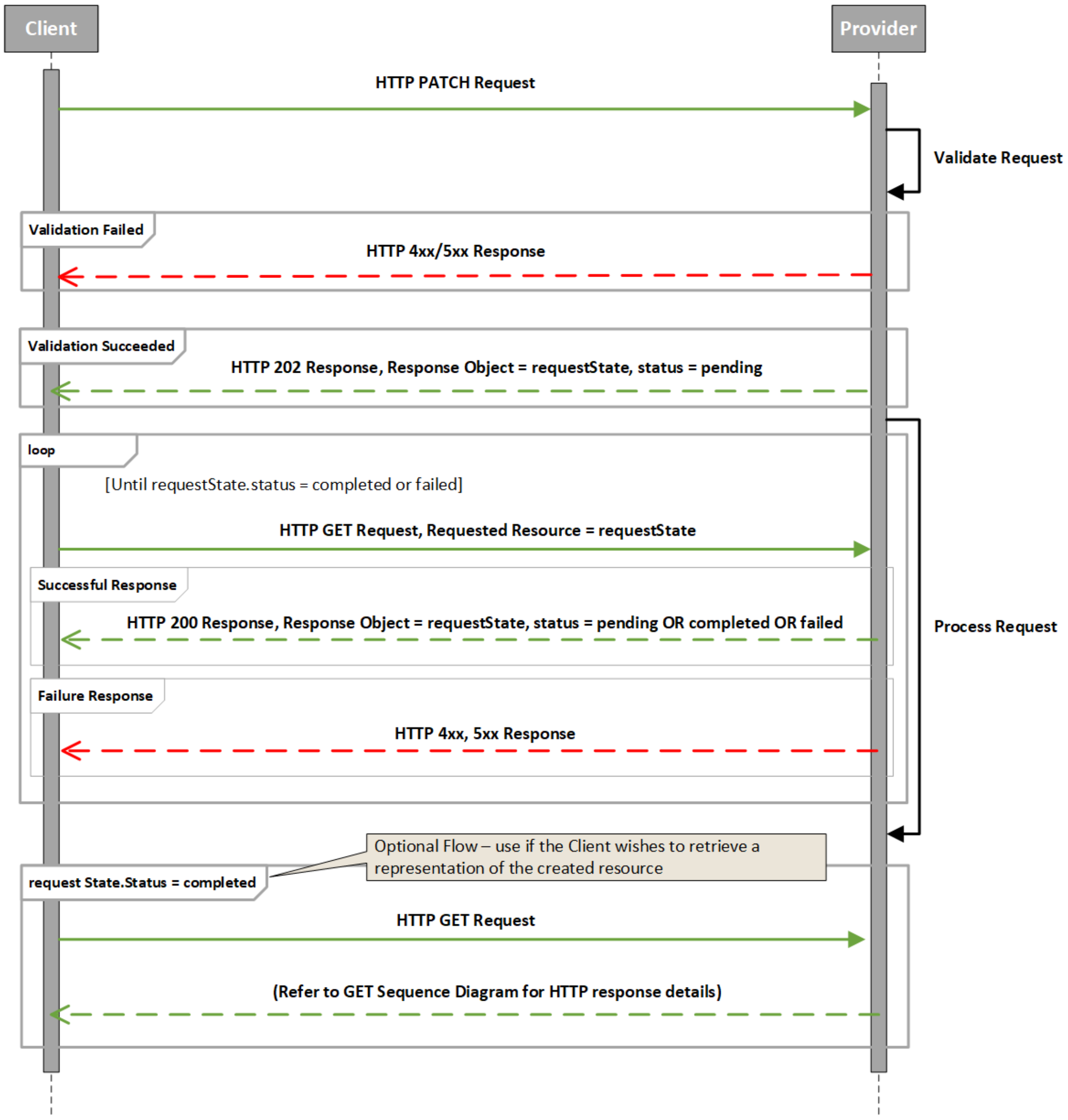
Client
Provider
HTTP PATCH Request
Validate Request
Validation Failed
HTTP 4xx/5xx Response
Validation Succeeded
HTTP 202 Response, Response Object = requestState, status = pending
loop
[Until requestState.status = completed or failed]
HTTP GET Request, Requested Resource = requestState
Successful Response
HTTP 200 Response, Response Object = requestState, status = pending OR completed OR failed
Process Request
Failure Response
HTTP 4xx, 5xx Response
Optional Flow – use if the Client wishes to retrieve a representation of the created resource
request State.Status = completed
HTTP GET Request
(Refer to GET Sequence Diagram for HTTP response details)

Figure 7: Mobile Money API Asynchronous Polling Diagram for Update Requests

3.2 API Error Handling

3.2.1 Summary of HTTP Response Codes

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

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

3.2.2 HTTP Error Response Codes

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
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

3.2.3 Errors Object Definition

The mobile money API uses the *errors* object to provide error details to API clients. The following properties are included:

- **Error Category**. All error codes must be associated with an appropriate category. This provides context and uniqueness to the error code.
- **Error Code**. Provides the reason for the request failure. Error codes are generally granular, i.e. they provide a specific reason for failure. In some cases, granularity is neither possible nor 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 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 Description**. A textual description of the error code.
- **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…

With the *errorParameters* property, care should be taken regarding confidentially of information. Confidential parameter information should only be disclosed to trusted clients.

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 Codes
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	Error Parameters	

3.2.3.1 Error Parameters Object

Allows error parameter properties to be specified in the form of key/value pairs. The number of key/value pairs is limited to 20.

Error Parameters Object Properties					
Name	**Type**	**Description**		**Reference**	**Validation**
key	string	Identifies the type of additional error parameter.	→M ←M		
value	string	Identifies the value of the additional error parameter.	→M ←M		

3.2.4 API Error Codes

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.
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.
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	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.

3.3 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 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		

3.4 Missing Response Retrieval

In some circumstances, the client may not have received the final representation of the resource for which it attempted to create. 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 Response Properties

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