Connector

For SWIFT gpi

Tracker API Specification

This document provides the description of the different APIs available on the gpi Tracker.

09 February 2018

Table of Contents

Preface **3**

1 gpi API Overview **4**

2 gpi Payment Transaction Data **6**

2.1 Payment Transaction 6
2.2 Payment Transaction Data 7

3 gpi API **13**

3.1 gpi API URL and RBAC Roles 13
3.2 gpi API Supported 14
3.3 gpi API Description 14
3.4 gpi API Failure 19
3.5 gpi API Detailed Specifications 22

A.1 Local Authentication Introduction **24**

A.2 Local Authentication Details **24**

B.1 Use of ApplAPIKey **27**

Legal Notices **28**

Preface

About this document

These technical specifications provide information about how to use Web services to access payment transaction information. The document describes the APIs in JSON format.

The purpose of this document is to provide the necessary information to assess the business use of the API by potential users of the API.

This version of the API technical specification incorporates the feedback given on the preliminary version.

Intended audience

This document is for the following audience:

- Business analysts to understand the functionality of the API
- Business architects to understand the functionality of the API and how to integrate the use of the API within their organisation
- Software developers of applications using the API to assess the complexity of the API

First edition

This is the first edition of the document.

Related documentation

- *Connector for SWIFT gpi Tracker API Detailed Specification*
- *Connector for SWIFT gpi Tracker API JSON Schemas*
- *Connector for SWIFT gpi Service Description*
- *SWIFT gpi Service Description*

1 gpi API Overview

Overall context view of the API

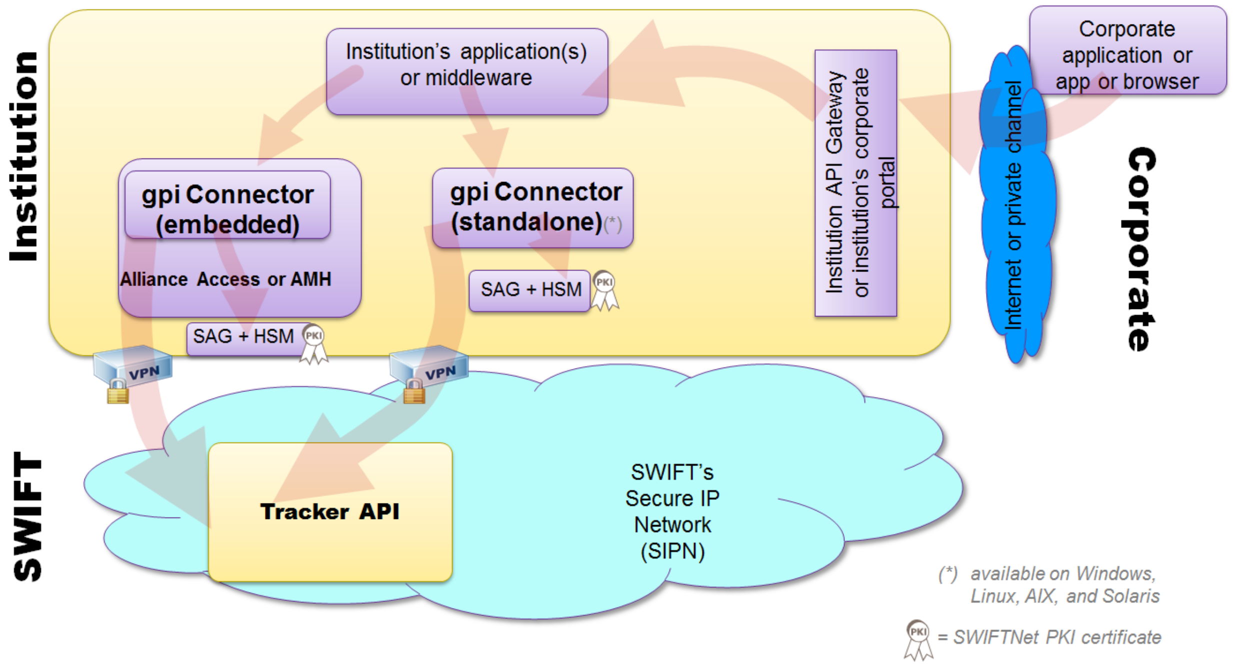
Institution
Institution's application(s) or middleware
gpi Connector (embedded)
Alliance Access or AMH
SAG + HSM
gpi Connector (standalone)(*)
SAG + HSM
Institution API Gateway or institution's corporate portal
Corporate application or app or browser
Internet or private channel
Corporate
VPN
VPN
SWIFT
Tracker API
SWIFT's Secure IP Network (SIPN)
(*) available on Windows, Linux, AIX, and Solaris
= SWIFTNet PKI certificate

Figure 1 - gpi Tracker API context diagram

Figure 1 shows the components that play a role when using the gpi APIs.

The components are the following:

- Institution's application(s) - this is the business application that requires an up-to-date information of a payment transaction involving gpi participants or that generate status confirmations
- Corporate application or browser - in case the institution integrates the use of gpi APIs into its web applications offered to its corporate customers.
- The SWIFT provided gpi Connector runs at the institution and exposes the gpi APIs, which means this component act as a proxy that manages security aspects: The gpi Connector can run in two ways
  - gpi Connector runs embedded in Alliance Access or Alliance Messaging Hub

    SAG/SNL + HSM
  - gpi Connector runs standalone using certificates on HSM

    SAG/SNL+HSM
- VPN box configured for SNL or the host running gpi Connector connecting the institution through the MV-SIPN
- The components running in SWIFT's Operating Centre (OPC) providing the Tracker API

Note *The information in this section is an introduction to configuration items or software deployments that are required to run the API. The information is provided to understand the overall picture of running the API, but is not needed to understand the purpose and use of the API itself.*

API Specification in this document

The gpi API exposed to the institution's applications is the same API as used towards the API Gateway running at SWIFT.

The SWIFT provided API component running at the institution is managing the security aspects to properly authenticate the applications initiating the gpi API. This mandatory authentication is based on HMAC using a symmetric key. This authentication mechanism is not specific for the API specified in this document, but is a generic requirement for all API services offered through SWIFT. Detailed information can be found in Local Authentication.

The API itself requires an API Key to identify the institution application that is using the API. The process to get the API Key is not further described in this document.

The API uses the SWIFTNet API framework that requires an authentication using the SWIFT Identity Services. The actual protocol of doing so is handled by the SWIFT provided API component and is not further described in this document.

REST design principles

The API follows REST design principles that provide simple and predictable URLs to access data.

API calls

HTTP requests use standard HTTP methods like GET, PUT, POST, and DELETE. The gpi APIs only use POST.

API responses

HTTP responses are UTF-8 encoded JSON objects.

gpi API live and pilot service

There is a live and a pilot API service. This is visible within the URLs to be used. For more information see *gpi API URL and RBAC Roles* and *gpi API Supported*.

Use of RBAC roles

The gpi API is only allowed when an appropriate RBAC (Role Based Access Control) role has been granted by the institution's security officer(s) to the certificate that is used by the SWIFT provided API component to authenticate itself using the SWIFT Identity Services. These roles are different for updating or for retrieving information.

The roles are different for pilot and for live services.

The same certificate can be used for live or for pilot API services. This is under full control of the security officer(s).

See *gpi API URL and RBAC Roles* for more information.

Multi-BIC support

The multi-BIC support is controlled through the appropriate RBAC role. This allows the same authenticated certificate to perform API calls for different BICs. See *gpi API URL and RBAC Roles* for more information.

Type of information retrieved by the API

The gpi API is used to get information on a payment transaction. The concept of a payment transaction and the different participants is described in *Payment Transaction*.

Type of information updated by the API

The only update is providing a status confirmation of a payment transaction.

2 gpi Payment Transaction Data

2.1 Payment Transaction

Payment transaction

Currently, a transaction is a serial flow of payment messages between an originator and a beneficiary.

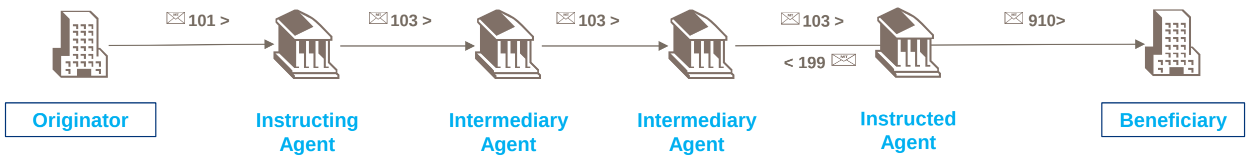

Figure 2 - Payment transaction

The figure shows 4 gpi banks participating in the payment transaction. The MT 103 exchanged between them carry 2 fields in the user header (block 3 of the FIN message), the field 111 containing the service type and the field 121 having the unique End-to-End transaction reference (UETR).

The transaction information of those MTs 103 is recorded into the gpi Tracker.

Similarly, the MT 199 containing the status updates contains the field 111 and 121 in the user header. The example flow contains only one MT 199 issued by the Instructed gpi Agent.

In the example flow in figure 2, the message coming from the originator (MT 101) and the message sent to the beneficiary (MT 910) are not recorded in the gpi Tracker.

gpi Agents

There are 5 different agents in the payment flow. These are:

- Ordering institution - the financial institution of the ordering customer. This can be a financial institution that subscribed to gpi or can be non-gpi financial institution.
- Instructing gpi Agent - the first gpi institution in the chain.
- Intermediary gpi Agent or intermediary institution - any subsequent gpi institution in the chain that forwarded the payment to the next one in the payment chain.
- Instructed gpi Agent - the last gpi institution in the payment chain. This is the institution that indicates the final status of the payment.
- Beneficiary institution or account with institution - the financial institution holding the account of the customer to be credited. This can be a financial institution that subscribed to gpi or can be non-gpi financial institution.

Tracking a payment

A payment transaction is one or more payment messages between gpi participants for which the status is reported through status confirmations.

Status updates can be MT 199 or a status update through an API.

How to know the status of a payment transaction

The status can be known by checking it using a SWIFT provided GUI where an entitled user can check the status of a payment transaction.

The status can also be known by using an API or by receiving a status confirmation from the Tracker if configured.

2.2 Payment Transaction Data

Overview of Payment Transaction Data

The payment transaction data for a given payment consists of all events identified by the same UETR

A payment transaction is a serial flow of payment messages (MT 103) sent between gpi institutions. The order of payment messages is based on the time when each payment message is sent.

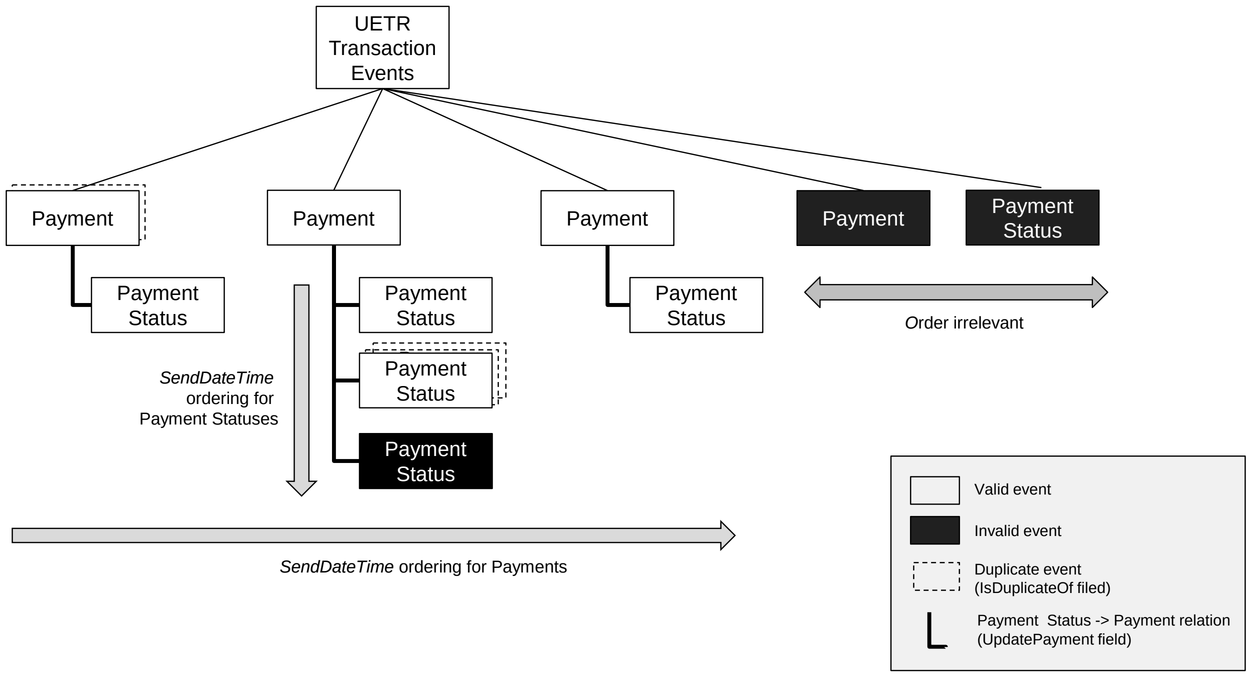

Figure 3 - Payment Transaction Data

Similarly, the status confirmation events are linked with a given payment message and if several updates are made for a given payment message these are also ordered in time of sending the update through an MT 199 or the time of update through the API.

Figure 3 also shows invalid events. These are events which are identified as being inconsistent or syntactically invalid. Invalid events can be categorised as follows:

- Inconsistent payment messages - these are payment messages that seem to be for a different transaction. As an example, the instructed amount is different. All status confirmation updates that are sent through an MT 199 for such invalid payment message are considered to be invalid as well. A status confirmation update for such inconsistent payment message sent by an API update is immediately nacked and not recorded in the Tracker.
- Inconsistent status confirmation updates - these are MTs 199 that are syntactically correct but whose status does not seem to be consistent with the transaction flow recorded. As an example, an intermediary agent sends a status confirmation update that the account is credited. Such status confirmation update through an API would be immediately nacked.

  It is unlikely but possible that a status confirmation update becomes invalid at a later time because of a subsequent update. This is only the case when a status confirmation update that made a payment transaction final is followed by a new valid payment sent with that UETR.
- Incorrectly formatted status confirmation updates - these are MTs 199 which contain a badly formatted status confirmation update. Such status confirmation updates cannot be assigned to a payment and are tracked immediately under the UETR. Such invalid formatted status confirmation update is not possible using the API since an invalid

status confirmation would be immediately nacked and not be stored. The behaviour of immediately nacking an incorrectly formatted status confirmation update may also become available for the MT 199.

Attributes of a transaction available in the API

The attributes of a transaction are related to the transaction, to a payment message or a status confirmation. Attributes reuse the ISO 20022 components available within the ISO 20022 repository. The attributes are described below in a table.

The table contains 4 columns:

- The name of the attribute
- A short description of the attribute
- The field in the MT 103, MT 199, or API from which the attribute is derived from
- An indication of the sensitivity of the attribute. This can be:
  - Public - anyone knowing the UETR can retrieve this attribute
  - Low - anyone participating in the transaction can retrieve this attribute
  - High - any gpi financial institution participating in the transaction can retrieve this attribute. For non-gpi financial institutions this may be returned as configured in the tracker. To retrieve such attribute a specific RBAC role is needed. See *gpi API URL and RBAC Roles* for more information.

2.2.1 Transaction Attributes

Description of attributes

Attribute	Description	MT field	Sensitivity
TransactionIdentification	Unique End-to-End transaction reference (UETR)	{3:{121:	Public
TransactionStatus	The status of transaction as reported in the last status confirmation of the last payment	:79:line2	Public
PreviousInstructingAgent	Ordering institution	:52A: If 52A in first payment absent then sender of payment	Low
CreditorAgent	Account with BIC	:57A:	Low
InitiationTime	Date+time in UTC based on SendDateTime of first Payment		Public
CompletionTime	Date+time in UTC based on SendDateTime of Status confirmation containing a final status ACSC		Public
InstructedAmount	Currency+Amount	:33B:	Low
ConfirmedAmount	Currency+Amount	:79:line4 or ConfirmedAmount of API	High
InterbankSettlementDate	Value date	:32A:	Low
LastUpdateTime	Date+time in UTC based on		Low

	SendDateTime of last Payment or Status confirmation		

2.2.2 Payment Message Attributes

Description of attributes

This table lists the attributes for the payment message. It is possible that some attributes are not present in case they are not present in the payment message.

Attribute	Description	MT field	Sensitivity
TransactionIdentification	Unique End-to-End transaction reference (UETR)	{3:{121:	Public
BusinessService	This qualifies the usage of the UETR and indicates what business rules apply	{3:{111:	Low
NetworkReference	Reference assigned by the network when sending the payment Primary Key	MIR	Low/High
MessageNameIdentification	Message type of the MT	{2:MessageType "103"	Low
Valid	Computed boolean indicating if a payment is consistent or not	Computed	Low
InvalidityReason	Code indicating why not valid	Computed	Low
InvalidityDescription	Textual description of the code	Computed	Low
SenderReference	Reference outside of the payload assigned by the sender of the payment. For MT format this is the MUR	{3:{108	Low
InstructionIdentification	Payload reference of the message	:20:	Low
From	BIC11 of the sender of the input message	{1:BIC	Low/High
To	BIC11 of the receiver of the input message	{2:BIC	Low/High
Orginator	Equal to the sender	{1:BIC	Low/High
PreviousInstructingAgent	The ordering institution	:52A:	Low
CreditorAgent	The financial institution having the account of the beneficiary	:57A:	Low/High
SenderAcknowledgementReceipt	DateTime in UTC at which the message was acked	FIN Ack {177:	Low/High

ReceivedDate	DateTime in UTC at which the message was acked by the receiver	Reception of FIN Ack as recorded by SWIFT	Low/High
InstructedAmount	Currency and amount as instructed by the ordering customer	:33B: :32A: if 33B absent	Low
InterbankSettlementAmount	Currency and amount of the interbank payment	:32A:	Low/High
InterbankSettlementDate	Value date of the interbank payment	:32A:	Low/High
ChargeBearer	Specification of which party will bear the charges for the transaction ("BEN", "OUR", or "SHA")	:71A:	High
ChargeAmount	List of repetitive field specifying the currency and amount of the transaction charges deducted by the sender and by previous banks in the transaction chain.	:71F:	High
ExchangeRate	Exchange rate	:36:	High
DuplicateMessageReference	Network reference of the payment that is the original recorded payment		Low
CopiedBusinessService	Used only when message is copied	{3:{103	Low

When the sensitivity is "Low/High" it means that the data is considered to be highly sensitive for non-gpi institutions as configured by the gpi institution. For instance, the ordering institution may not see charges and routing related information if its gpi institution decides so.

2.2.3 Status Confirmation Attributes

Description of attributes

This table lists the attributes from the status confirmation. It is possible that some attributes are not present in case they are not present in the status confirmation.

Attribute	**Description**	**MT field**	**Sensitivity**
TransactionIdentification	Unique End-to-End transaction reference	{3:{121:	Public
BusinessService	This qualifies the usage of the UETR and indicates what business rules apply	{3:{111:	Low
NetworkReference	Reference assigned by the	MIR for MT199 NetworkReference from	Low/High

	network when sending the payment Primary Key	UpdatePaymentStatus API	
MessageNameIdentification	Message type of the MT	{2:MessageType "199" Or "camt.a01.001.02" for API	Low
Valid	Computed boolean indicating if a status confirmation is consistent or not	Computed	Low
InvalidityReason	Code indicating why not valid	Computed	Low
InvalidityDescription	Textual description of the code	Computed	Low
SenderReference	Reference outside of the payload assigned by the sender of the payment. For MT format this is the MUR	{3:{108 NULL in case of API	Low
InstructionIdentification	Payload reference of the message	:20: for MT199 InstructionIdentification from UpdatePaymentStatus API	Low
TransactionStatus	Status of the transaction	:79:line 2 or TransactionStatus from UpdatePaymentStatus API	Public
ForwardedAgent	The BIC to which the originator has forwarded the payment transaction	:79:line 3 "forwarded to" part or ForwardedAgent from UpdatePaymentStatus API	Low/High
From	11-character BIC of the sender of the input message or 8-character BIC issuing the API call	{1:BIC for MT199 or 8-character BIC of API	Low/High
Orginator	BIC11 reporting the status update	:79:line3 or Originator of API	Low/High
FundsAvailable	DateTime in UTC as reported in the ACSC status update	:79:line1 or FundsAvailable of API	Low
SenderAcknowledgementReceipt	DateTime in UTC at which the message was acked	FIN Ack {177: Or DateTime of reception of UpdatePaymentStatus API	Low
ConfirmedAmount	Currency and amount confirmed by originator	:79:line4 or ConfirmedAmount of API	High
ChargeAmount	List of repetitive field specifying the currency and amount of the transaction charges deducted	:79: lines containing //:71F: Or ChargeAmount in UpdatePaymentStatus API	High

	by the sender and by previous banks in the transaction chain.		
ForeignExchangeDetails	Exchange rate	:79: line containing //EXCH Or ForeignExchangeDetails in UpdatePaymentStatus API	High
UpdatePayment	Contains the network reference of the payment for which the status confirmation was applicable		Low/High
DuplicateMessageReference	Network reference of the payment that is the original recorded Payment		Low

When the sensitivity is "Low/High" it means that the data is considered to be highly sensitive for non-gpi institutions as configured by the gpi institution. For instance, the ordering institution may not see charges and routing related information if its gpi institution decides so.

3 gpi API

3.1 gpi API URL and RBAC Roles

How the URL identifies the live or test services

The URL used for each API consists of 2 items. One is the specific domain to be used in order to reach the SWIFT provided component within the financial institution and the remaining is the actual API call.

For live data, the URL looks like:

```
https://local-api-domain/swift-apitracker/v1/api-specific-part
```

For test data, the URL looks like:

```
https://local-api-domain/swift-apitracker-pilot/v1/api-specific-part
```

Note *The exact name of the local-api-domain is not specified in this document. It will be described within the SWIFT provided component documentation.*
The SWIFT provided component maps the local-api-domain to the public domain `apigtw.swiftnet.sipn.swift.com`

The API-specific part

The API specific part consists of an indication of the resource it is managing and additional parameters. In the gpi API Description section, the API-specific part of the URL is provided per API.

RBAC roles

SWIFT uses RBAC roles to control access to the payment transaction data. The main purpose is to identify what privileges are given by the security officer(s) of the institution to the certificate used to connect to SWIFT. This certificate is managed by the local SWIFT component but the roles are managed by the institution's security officers.

There are currently 3 RBAC roles foreseen for live and 3 RBAC roles for test.

The roles are:

- Update - used to control who can perform a status update for a payment.
- StandardViewer - used to control who can get the low sensitive information for a payment transaction.
- FullViewer - used to control who can get all information for a payment.

The scope of each of these roles is for a given business party identifier. This is indicated by qualifying the role name with the scope of a given BIC. Roles are therefore triples.

```
RoleName/Scope/BIC8
```

Examples are:

```
Update/Scope/bankbeb0

StandardViewer/Scope/zyacnl20

FullViewer/Scope/userus33
```

In these examples, the first two are for the test API service and the last for the live API service. This is because the two first have a Test & Training BIC and the last a live BIC.

It is possible to assign to the same certificate roles of a different business party identifier on condition that this has been provisioned by SWIFT. This provisioning is not described further in this document.

See gpi API Description for more details on how to specify in the API what RBAC roles are applicable for a given API invocation.

Granting RBAC roles

This is done by the security officer(s) just like any other RBAC role. The result of the granting of the roles is that a given certificate used to establish a session with the API Gateway can issue API requests for other BICs.

Note *The service names are `swift.apitracker` for live and `swift.apitracker!p` for test. These service names are visible to the security officer(s) when assigning the RBAC roles. A simple convention is used to derive the URL used, namely replace the "." by a "-" and "!p" by "-pilot".*

3.2 gpi API Supported

Current APIs

The APIs that are targeted for the first release are all related to payment transactions.

- POST …/status_confirmations: Updating the status of a payment transaction
- POST …/get_payment_transaction_details: Getting a payment transaction
- POST …/get_payment_transactions: Searching for payment transactions
- POST …/get_changed_payment_transactions: Getting the history of payment transactions
- POST …/get_invalid_events: Getting invalid events

These 5 API are described in *gpi API Description*.

Future APIs

New APIs may be identified to cover the need of applications handling the current payment transaction data as described in *gpi Payment Transaction Data*.

New APIs are required in case the payment transaction data is extended to other type of data, for instance to supporting documentation of a payment transaction. Such APIs may be the appropriate way to handle such data flows.

3.3 gpi API Description

Note *The JSON structure will be added in the final version of the document. However, the information in the request and response attributes is sufficient to understand the functionality offered by the API.*

Specifying RBAC roles in the API

The designer of the application calling the gpi API can control the scope of a given API to be restricted to a given BIC or list of BICs.

The instruction is within a dedicated HTTP header. To ensure that the application exercises this control, this HTTP header is mandatory.

As an example, for the status update API the application calling the API needs to indicate for what BIC the update API is invoked. This is done through adding an `RBACRole` HTTP header as follows:

```
RBACRole: [Update/Scope/zyacgb20]
```

This example is for the test service since the BIC is a Test and Training BIC. It shows that the role is a triple separated by "/". The "[" and "]" brackets are mandatory.

In case more than one BIC is in scope, then those are enumerated as follows:

```
RBACRole: [Update/Scope/zyacgb20/zyacnl20]
```

Note *This field is signed locally between the application generating the API and the gpi Connector. See for more information Local Authentication.*

Filtering attributes on responses

When applicable, a table is added to explain what filtering is done on the response. This filtering depends on the role the BIC is playing within the transaction, and, whether the BIC is a gpi participant.

The table looks like:

gpi?	Not participating	Ordering BIC	Instructing gpi Agent	Intermediary gpi Agent	Instructed gpi Agent	Beneficiary BIC
Yes						
No						

The first column "gpi?" indicates if the BIC in the RBACRole is a gpi participant or not.

The second column "Not participating" applies when the BIC is not an agent in the payment transaction. The different agent roles are described in Payment Transaction. When this column contains "Not applicable", it indicates that the payment transaction is not applicable for the API response and therefore nothing is returned for that payment transaction.

The last 5 columns are the different agents that can participate in a payment transaction.

3.3.1 Payment Transactions: Updating the Status of a Payment Transaction

Purpose of the API

This API is a status confirmation update to inform the Tracker about the updated status of a given payment.

RBAC role required

The HTTP header requires a field specifying the RBAC role to be checked. This is the business party identifier for which the status update is done.

```
RBACRole: [Update/Scope/bankbeb0]
```

There is only one such field and one role to be specified.

Request URI

POST …/status_confirmations

Request/response body structure and elements

See "UpdatePaymentStatus" API in *Connector for SWIFT gpi Tracker API Detailed Specification*.

3.3.2 Payment Transactions: Getting a Payment Transaction

Purpose of the API

This API is a payment query to get detailed information regarding a given payment. It requires the UETR to be known.

Examples of use cases are:

- Debtor status check - ordering BIC or ordering customer wants to see the status
- Creditor status check - beneficiary BIC or beneficiary customer wants to see the status

RBAC role required

The HTTP header requires a field specifying the RBAC role to be checked. There are 2 possible roles that can be used as shown in the table:

RBAC role	Description
[StandardViewer/Scope/bankbeb0]	Low sensitive data is requested for a specific BIC to be checked

	to be in scope of the certificate
[FullViewer/Scope/bankbeb0]	Low and high sensitive data is requested for a specific BIC to be checked to be in scope of the certificate

Since the application is typically not aware about what roles are assigned to a given certificate, both roles should be put as follows:

```
RBACRole: [StandardViewer/Scope/bankbeb0] [FullViewer/Scope/bankbeb0]
```

In case multiple BIC are used these should be enumerated by adding additional BIC separated by a "/" as follows:

```
RBACRole: [StandardViewer/Scope/bankbeb0/usrvgb20]
[FullViewer/Scope/bankbeb0/usrvgb20]
```

The API succeeds when at least one of the roles is granted for all BICs enumerated to the certificate.

Request URI

POST …/get_payment_transaction_details

Request/response body structure and attributes

See "GetPayment" API in *Connector for SWIFT gpi Tracker API Detailed Specification*.

The response attributes are filtered according to the BIC in the request. The following table gives the sensitivity of the data returned depending on the type of BIC:

gpi?	Not participating	Ordering BIC	Instructing gpi Agent	Intermediary gpi Agent	Instructed gpi Agent	Beneficiary BIC
Yes	Public	RBAC driven: High if FullViewer RBAC role is granted to the certificate Low if StandardViewer RBAC role and no FullViewer role is granted				
No	Public	RBAC driven as above and additionally restricted to Low/High as decided by Instructing gpi Agent				RBAC driven as above and additionally restricted to Low/High as decided by Instructed gpi Agent

3.3.3 Payment Transactions: Searching for Payment Transactions

Purpose of the API

This API is a payment transaction search to get transaction-level information regarding all payments that match the search criteria. To have full information of a given payment transaction the API defined in Payment Transactions: Getting a Payment Transaction can be used.

Examples of use cases are:

- Investigation by a participating institution based on criteria given by a customer, such as instructed amount, time of initiation
- Flow monitoring, for instance to find payments that are "stuck" for which the institution is the beneficiary institution
- Feed a dashboard - give all payment transactions with their last status

RBAC role required

The HTTP header requires a field specifying the RBAC role to be checked. There are 2 possible roles that can be used as shown in the table:

RBAC role	Description
[StandardViewer/Scope/bankbeb0]	Low sensitive data is requested for a specific BIC to be checked to be in scope of the certificate
[FullViewer/Scope/bankbeb0]	Low and high sensitive data is requested for a specific BIC to be checked to be in scope of the certificate

Since the application is typically not aware about what roles are assigned to a given certificate, both roles should be put as follows:

```
RBACRole: [StandardViewer/Scope/bankbeb0] [FullViewer/Scope/bankbeb0]
```

In case multiple BIC are used these should be enumerated by adding additional BIC separated by a "/" as follows:

```
RBACRole: [StandardViewer/Scope/bankbeb0/usrvgb20]
[FullViewer/Scope/bankbeb0/usrvgb20]
```

The API succeeds when at least one of the roles is granted for all BICs enumerated to the certificate.

Request URI

POST …/get_payment_transactions

Request/response body structure and attributes

See "GetPaymentTransaction" API in *Connector for SWIFT gpi Tracker API Detailed Specification*.

The response attributes are filtered according to the BIC in the request. The following table gives the sensitivity of the data returned depending on the type of BIC:

gpi?	Not participating	Ordering BIC	Instructing gpi Agent	Intermediary gpi Agent	Instructed gpi Agent	Beneficiary BIC
Yes	Not applicable	RBAC driven: High if FullViewer RBAC role is granted to the certificate Low if StandardViewer RBAC role and no FullViewer role is granted				
No	Not applicable	RBAC driven as above and additionally restricted to Low/High as decided by Instructing gpi Agent				RBAC driven as above and additionally restricted to Low/High as decided by Instructed gpi Agent

3.3.4 Payment Transactions: Getting the Last Changed Payment Transactions

Purpose of the API

This API is a delta query to get all payment update information starting from a given date and time. This API allows synchronization of a local database with the Tracker database. There is a paging mechanism in case the response becomes too large. This is done through returning a "More" token in the response, so that the delta query can continue from the last returned result.

All use cases applicable for the get and search of payment transactions apply also for this API.

The use of the API to get the last changed payment transactions is driven by an architectural choice to feed a local database and to optimise the flow between the Tracker and the API caller.

RBAC role required

The HTTP header requires a field specifying the RBAC role to be checked. This is the FullViewer role for the given business party identifier as follows:

```
RBACRole: [FullViewer/Scope/bankbeb0]
```

In case multiple BIC are used these should be enumerated by adding additional BIC separated by a "/" as follows:

```
RBACRole: [FullViewer/Scope/bankbeb0/usrvgb20]
```

The API succeeds when the role is granted for all BICs enumerated to the certificate.

Request URI

POST …/get_changed_payment_transactions

Request/response body structure and attributes

See "GetChangedPaymentTransactions" API in *Connector for SWIFT gpi Tracker API Detailed Specification*.

The response attributes are filtered according to the BIC in the request. The following table gives the sensitivity of the data returned depending on the type of BIC:

gpi?	Not participating	Ordering BIC	Instructing gpi Agent	Intermediary gpi Agent	Instructed gpi Agent	Beneficiary BIC
Yes	Not applicable	RBAC driven: this API requires a FullViewer role				
No	Not applicable	RBAC driven as above and additionally restricted to Low/High as decided by Instructing gpi Agent				RBAC driven as above and additionally restricted to Low/High as decided by Instructed gpi Agent

3.3.5 Payment Transactions: Getting Invalid Events

Purpose of the API

This API is a payment events' query to get invalid events for payment transactions within a given time frame. An event is either a payment or a status confirmation.

The API is typically used for support or audit purposes, where all invalid messages sent or received are retrievable by one API call.

RBAC role required

The HTTP header requires a field specifying the RBAC role to be checked. This is the FullViewer role for the given business party identifier as follows:

```
RBACRole: [FullViewer/Scope/bankbeb0]
```

In case multiple BIC are used these should be enumerated by adding additional BIC separated by a "/" as follows:

`RBACRole: [FullViewer/Scope/bankbeb0/usrvgb20]`

The API succeeds when the role is granted for all BICs enumerated to the certificate.

Request URI

POST …/get_invalid_events

Request/response body structure and attributes

The response attributes are filtered according to the BIC in the request. Since the FullViewer role was checked for that BIC, all data of the invalid event as recorded is returned. This may include High sensitive information as well.

See "GetInvalidEvents" API in *Connector for SWIFT gpi Tracker API Detailed Specification*.

3.4 gpi API Failure

Structure of a status indicating a failure

At failure, all previously described APIs shall return the following "Status" structure instead of the successful response body:

Attribute	Format	Description
severity	Text	The severity of the error (Transient, Fatal, Logic)
code	Text	The error code
text	Text	The error message targeting the consumer application

Using the "severity"

When the severity is "Transient" then a retry of the same API may work, depending whether the condition that resulted in the error has been resolved. The retry should be paced and not be done longer than 5 minutes.

When the severity is "Fatal" then this indicates something wrong with the API. A retry will fail again. In some cases the error is related to configuration that is not related to the service for which the API is done.

When the severity is "Logic" then this indicates a protocol violation of some sort that can be resolved by some action such as issuing another API. The expected behaviour is documented as part of some product or vendor documentation. If the error is not recognized and the expected behaviour thus not implemented then the retry is done as for Transient errors.

Errors related to gpi processing

The following errors may be returned by gpi.

Error Description	HTTP Response Code	JSON Object

Syntax error	400 – Bad Request	{“status”: {
		“severity”: “Fatal”,
		“code”: “X001”,
		“text”: “Syntax error” }}

Status originator not allowed to update status of this payment.	400 – Bad Request	{“status”: { “severity”: “Fatal”, “code”: “X002”, “text”: “Status originator not allowed to update status of this payment” }}
Status update of payment message not allowed because already in final state	400 – Bad Request	{“status”: { “severity”: “Fatal”, “code”: “X003”, “text”: “Status update of payment message not

Error Description	HTTP Response Code	JSON Object
		allowed because already in final state" }}
No payment information tracked for this transaction identifier. Unknown transaction identifier	400 – Bad Request	{"status": { "severity": "Fatal", "code": "X004", "text": "No payment information tracked for this transaction identifier. Unknown transaction identifier" }}
Status update sent by institution that does not participate in transaction	400 – Bad Request	{"status": { "severity": "Fatal", "code": "X008", "text": "Status update sent by institution that does not participate in transaction" }}

Invalid request	400 – Bad Request	{"status": {
		"severity": "Fatal",
		"code": "Sw.gpi.InvalidRequest",
		"text": "Invalid request" }}

Mandatory field missing	400 - Bad Request	{"status": {
		"severity": "Fatal",
		"code": "Sw.gpi.MandatoryFieldMissing",
		"text": "Mandatory field %field missing" }}

MyInstitution authorization failure	401 - Unauthorized	{"status": { "severity": "Fatal", "code": "Sw.gpi.MyInstitutionAuthorizationFailure", "text": "MyInstitution %MyInstitution does not have the appropriate RBAC role" }}
From authorization failure	401 - Unauthorized	{"status": { "severity": "Fatal", "code": "Sw.gpi.FromAuthorizationFailure", "text": "From %From does not have the appropriate RBAC role" }}
No result found	404 - Not Found	{"status": { "severity": "Transient", "code": "Sw.gpi.NoResultFound", "text": "Request did not return any result. This may be a transient condition" }}
Unknown transaction	400 - Bad Request	{"status": { "severity": "Fatal", "code": "Sw.gpi.UnknownTransaction", "text": "Transaction is not known." }}

Invalid RBAC role	400 - Bad Request	{"status": {
		"severity": "Fatal",
		"code": "Sw.gpi.InvalidRBACRole",
		"text": "RBAC Role" }}

Internal error	500 – Internal Server Error	{"status": {
		"severity": "Transient",
		"code": "Sw.gpi.InternalError",
		"text": "Internal error" }}

Note *New errors can be added reflecting enhancements implemented, or, existing errors may be changed to better describe the error condition it is reporting upon.*

Errors related to the API framework

These error codes may be returned by SWIFT. These error codes are listed below:

Error Description	HTTP Response Code	JSON Object

API service not provisioned	401 – Unauthorized
	WWW-Authenticate: Bearer realm=”SwAP”
		{“status”: {
		“severity”: “Fatal”,
		“code”: “SwAP001”,
		“text”: “API service not provisioned” }}

Requestor has no role for service	401 – Unauthorized WWW-Authenticate: Bearer realm=”SwAP”	{“status”: { “severity”: “Logic”, “code”: “SwAP002”, “text”: “Requestor has no role for service” }}

Authentication failure	401 – Unauthorized
	WWW-Authenticate: Bearer realm=”SwAP”
		{“status”: {
		“severity”: “Fatal”,
		“code”: “SwAP003”,
		“text”: “Authentication failure” }}

Missing or Invalid API key	401 – Unauthorized
	WWW-Authenticate: Bearer realm=”SwAP”
		{“status”: {
		“severity”: “Fatal”,
		“code”: “SwAP005”,
		“text”: “Missing or Invalid API key” }}

Service quota exceeded	429 – Too many requests	{“status”: {
		“severity”: “Transient”,
		“code”: “SwAP006”,
		“text”: “Service quota exceeded” }}

System quota exceeded	429 – Too many requests	{“status”: {
		“severity”: “Transient”,
		“code”: “SwAP007”,
		“text”: “System quota exceeded” }}

Bad response received from service provider	502 – Bad Gateway	{“status”: { “severity”: “Transient”, “code”: “SwAP008”, “text”: “Invalid response received from Service Provider” }}
Service provider is temporarily unavailable	503 – Service Unavailable	{"status":{ "severity": "Transient", "code": "SwAP010", "text": "Service provider is temporarily unavailable"}}

Service provider time out	504 – Gateway Timeout	{“status”: {
		“severity”: “Transient”,
		“code”: “SwAP009”,
		“text”: “Service Provider time out” }}

One of the roles listed is not valid or none of the requested roles are granted to user.	401 – Unauthorized
	WWW-Authenticate: Bearer realm=”SwAP”
		{"status":{
		"severity": "Logic",
		"code": "SwAP011",
		"text": "Invalid HTTP RBAC header"}}

Internal SwAP API Gateway error	500 – Internal Server Error	{“status”: {
		“severity”: “Transient”,
		“code”: “SwAP099”,
		“text”: “Internal SwAP API Gateway error” }}

ADC WAF violation	401 – Unauthorized	{"status":{

Error Description	HTTP Response Code	JSON Object
	WWW-Authenticate: Bearer realm="SwAP"	"severity": "Fatal", "code": "SwAP090", "text": "There was an error while processing the request. Please consult SWIFT support and mention the following code <%TS.request.ID()%>"}}

Note *New errors can be added reflecting enhancements implemented, or, existing errors may be changed to better describe the error condition it is reporting upon.*

3.5 gpi API Detailed Specifications

Overview

The detailed specifications can be found in the document *Connector for SWIFT gpi Tracker API Detailed Specification*. This document is generated similar to the documentation for ISO 20022 messages.

The ISO 20022 data dictionary is used and the same data definitions are taken as much as possible. This chapter contains additional restrictions applicable to the gpi API.

3.5.1 Additional Restrictions

Overview

This section summarizes some additional restrictions applicable on the requests for the APIs.

8-character BIC versus 11-character BIC

MyInstitution is limited to 8-character BIC only. This is because the RBAC role used to validate this field is also limited to 8-character BIC only.

GMT or Zulu time only

All times are Zulu time only.

3.5.2 gpi Specific Code List

Overview

The invalidity status as discussed in Payment Transaction Data is returned within the API in the `InvalidityReason` and `InvalidityDescription`. These codes are an external code list following the ISO 20022 format for code lists. The possible values can be found in this table

Code	Description
X001	Syntax error.
X002	Status originator not allowed to update status of this payment.
X003	Status update of payment message not allowed because already in final state.
X004	No payment information tracked for this transaction identifier. Unknown transaction identifier.
X005	Instructed amount of payment message does not match previously sent payment messages for the same transaction identifier.
X006	Payment message does not match previously sent payment messages for the same transaction identifier.
X007	Payment message tracked that invalidates the status

	update.
X008	Status update sent by institution that does not participate in transaction.
X009	PreviousInstructingAgent in payment message does not match previously sent payment messages for the same transaction identifier.

Appendix A Local Authentication

A.1 Local Authentication Introduction

Local Authentication concept

Local authentication is a simple protocol between two parties that allows the authentication of one party to the other by means of a signature based on a shared secret symmetric key.

In our case, the parties are the application that generates the API request and the gpi Connector that will forward the request to SWIFT once that request has been properly authenticated.

Local Authentication algorithm

The algorithm is HMAC-SHA-256, as described in ISO/IEC 9797.

The key is at least 128 bits long.

The message authentication code is the first 128 bits of the result of the algorithm.

Local Authentication keys

Local authentication keys have to be configured at both parties. They must be stored securely, using access control and encryption. The local authentication keys must be unpredictable, for instance by using a pseudo random generator.

The gpi Connector uses a 4-eyes principle where the Local Authentication key consists of a left and a right part, which are configured separately by two operators (also known as left security officer and right security officer).

What is protected by the Local Authentication signature?

Besides some HTTP entity header fields as described below that are ensuring that a signature cannot be easily replayed, the following data is signed:

- The API specific part of the URI for the request
- The body of the HTTP request or response

A.2 Local Authentication Details

Local Authentication on API

The Local Authentication uses a set of dedicated HTTP entity header fields. They all start with LAU.

The current implementation foresees following headers:

- LAUApplicationID: ID that identifies the application generating the API and used by the gpi Connector to retrieve the related LAU keys.
- LAUVersion: version of the LAUSigned header. Mandatory. "1.0" for this first release.
- LAUCallTime: timestamp in UTC of the API call in the format YYYY-MM-DDTHH:MM:SS.sssZ. As an example `2016-11-30T09:22:45.321Z` Mandatory. The gpi Connector will compare its value (that must be a timestamp) with the actual time and will reject the request if the difference is more than a predefined value (for example, 5 minutes) in the past or in the future. That timestamp validation is a protection against replay attacks.
- LAURequestNonce: a random value generated by the client. Provided with the request and copied by the gpi Connector on the response.
- LAUResponseNonce: a random value generated by the gpi Connector on the response.
- LAUSigned: service specific HTTP headers
- LAUSignature: contains the LAU signature , base64 encoded.

LAU header field presence

The table shows what LAU headers are to be added by the application.

LAU Header field	Request	Response
LAUApplicationID	1..1	Equal to the Request
LAUVersion	1..1	Equal to the Request
LAUCallTime	1..1	1..1
LAURequestNonce	1..1	Equal to the Request
LAUResponseNonce	Forbidden	1..1
LAUSigned	1..1 for gpi	0..1 (but for gpi not present)
LAUSignature	1.1	1..1

LAUApplicationID

The LAUApplicationID contains the name of the application that generates the API. This name is configured within the gpi Connector. The configuration in the gpi Connector includes parameters that control how the TLS 1 way session is set up between the application and the gpi Connector.

LAUVersion

The LAUVersion is currently "1.0". It may be changed when new security algorithms are implemented or new LAU specific headers are added. This version is not indicating the version of the API that is exchanged using the gpi Connector.

Note *SWIFT reconsiders the algorithms and key lengths used by SWIFTNet at regular intervals. Vendors should be aware that SWIFT may advise changes to them following a suitable notice period.*

LAUCallTime

The LAUCallTime is set by the client application having the LAU key to the current time in UTC. This requires that system times are kept up to date.

The LAUCallTime is the defense against the replay of copied requests at a later time. The short period in which the LAU result remains useable allows a reasonable difference in the setting of the time between different systems initiating the API call.

LAURequestNonce

The LAURequestNonce ensures that each response has a different signature. Indeed, the LAURequestNonce is replayed in the response and signed. Additionally it ensures that each request has a different signature in case the LAUCallTime would be the same.

The format of the LAURequestNonce is left open, as long as it is the result of a pseudo random generated value.

LAUResponseNonce

This is the defense used by the gpi Connector against signing the same response. The LAUResponseNonce is generated by the gpi Connector using a pseudo random generator available in Java.

LAUSigned

The LAUSigned header field contains the signed headers that are passed to SWIFT in the request.

The service specific header fields that are to be provided are

- ApplAPIKey
- RBACRole

To respect the HTTP header format, the ApplAPIKey and RBACRole headers are grouped by comma separated headers as follows

“(” HTTPHeaderName “=” HTTPHeaderValue “)” [“,(” HTTPHeaderName “=” HTTPHeaderValue “)”]*.

As an example:

```
LAUSigned: (ApplAPIKey=ddfb7833-58eb-4ed2-ae2e-13bc521f5731),(RBACRole=
[FullViewer/Scope/bankbeb0])
```

Note *Within the response, such headers should be provided to the application. For the gpi API all business data is in the message body. Therefore, the LAUSigned is not present within the response.*

LAUSignature

The LAUSignature field contains the result of the HMAC algorithm using the LAU key on the data to be signed.

The data signed on the request

The data to be signed is for the request is the concatenated string of the following

- All HTTP headers fields starting with "LAU", ordered in lexicographic order, and where the leading and trailing spaces and tabs are removed. The HTTP header itself including the ":" and the CRLF at the end of the HTTP header field are included.
- The abs_path and the optional query part of the request URI including the CRLF at the end. In case the abs_path is absent, then an "/" is used instead.

Note *This data is the URL without the domain specific part and optional port number.
The terminology abs_path is coming from RFC 2616. This specifies within section 3.2.2 http URL the syntax of the URL as follows:
http_URL = "http:" "//" host [":" port] [abs_path ["?" query]]
What is signed is abs_path ["?" query]
Example: if the URL is equal to
https://example.com:4546/swift.apitracker/v1/api-specific-part?query
then the data signed is
/swift.apitracker/v1/api-specific-part?queryCRLF
with CRLF the control characters added at the end.*

- The message body, where the CRLF separator that separates the HTTP header from the message body is not included. If there is no message body, then only the HTTP headers starting with "LAU" and the request URI part are signed.

The data signed on the response

The data to be signed is for the response is the concatenated string of the following

- All HTTP header fields starting with "LAU", ordered in lexicographic order, and where the leading and trailing spaces and tabs are removed. The CRLF at the end of the HTTP header field is included.
- The message body, where the CRLF separator that separates the HTTP header from the message body is not included.

Verification of a signature

When verifying a signature the same data has to be used as input to the HMAC algorithm.

This means that the HTTP header field LAUSignature has to be excluded from the HTTP header fields starting with "LAU".

Otherwise the same process as for signing has to be followed.

Appendix B Use of ApplAPIKey

B.1 Use of ApplAPIKey

Concept of APIKey

An APIKey is used to identify an application generating an API to the server consuming the API.

There are two components involved in sending the API, namely the gpi Connector and the application generating the API.

Therefore there will be two APIKey within the request received by SWIFT: the APIKey identifying the gpi Connector and the ApplAPIKey identifying the application generating the API.

APIKey for the gpi Connector

This APIKey is known and inserted in the API request by the gpi Connector. Its value is checked by SWIFT.

ApplAPIKey for the application

This ApplAPIKey is to be registered to be used for gpi by the developer of the application calling the API.

The ApplAPIKey is in the form of an UUID as specified in IETF's RFC 4122. The format is therefore xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx in lower case.

The ApplAPIKey is generated by the application vendor using the gpi APIs and must be used within each API request as part of the LAUSigned HTTP header field.

The ApplAPIKey will be checked by SWIFT at a later date. Further details about the registration process will be provided at a later stage.

Legal Notices

Copyright

SWIFT © 2018. All rights reserved.

Disclaimer

SWIFT supplies this publication for information purposes only. The information in this publication may change from time to time. You must always refer to the latest available version.

Translations

The English version of SWIFT documentation is the only official and binding version.

Trademarks

SWIFT is the trade name of S.W.I.F.T. SCRL. The following are registered trademarks of SWIFT: the SWIFT logo, SWIFT, SWIFTNet, Sibos, 3SKey, Innotribe, the Standards Forum logo, MyStandards, and SWIFT Institute. Other product, service, or company names in this publication are trade names, trademarks, or registered trademarks of their respective owners.