WHITE PAPER

BOX Instant Payments Implementation and Concept

Version 3 Release 23 (V3R23)

Box Messaging Hub Instant Payments Implementation and Concept

Revision 3

INTERCOPE

Table of Contents

List of Tables

Table of Graphs

1 Introduction

1.1 Instant Payments

Instant payments – also known as real-time or immediate payments – are defined by the Euro Retail Payments Board (ERPB) as electronic retail payments that are available 24/7/365. They require the immediate or close-to-immediate interbank clearing of the transaction and crediting of the payee's account with confirmation to the payer (usually within a maximum of 10 seconds of payment initiation).

Instant payment focusses on low value retail payment systems (RPS); which differ from real-time gross settlement systems (RTGS) and distributed ledger payment systems.

Instant payments systems tend to have the following characteristics:

Immediate Credit

The funds become available in the payee's account immediately (within a few seconds) of the payment being initiated by the payer.

Irrevocability

Once the payer has initiated the payment, the payment process cannot be cancelled.

Certainty of Fate

When the payer initiates the payment, he/she is informed immediately (within a few seconds) whether the payment has successfully reached the payee's account or not.

The following graph gives an overview of what is achieved by Instant Payments and which components are generally involved in the transaction.

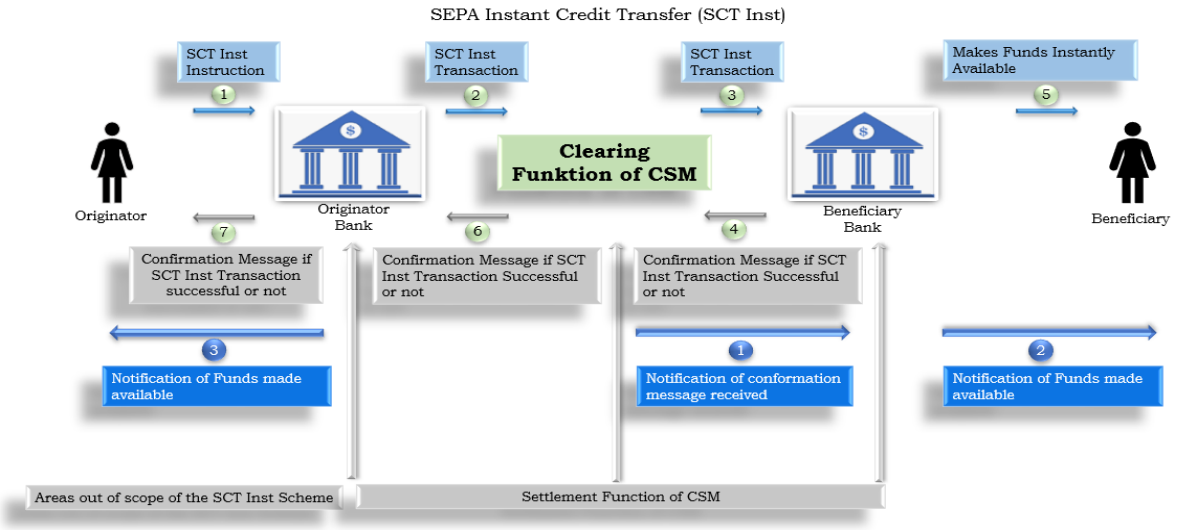


Figure 1 SEPA Instant Credit Transfer /SCT Inst) Overview

This document is designed and written to outline the BOX Messaging Hub (BOX) Instant Payment concept and implementation for SEPA Credit Transfer (SCT) Instant Payments scheme in Europe.

2 Architecture

2.1 BOX, the Messaging Hub

The BOX represents a financial gateway and Messaging Hub integrating with back-office systems and multiple networks as illustrated below in the graph. BOX is a multi-network solution. Within that, BOX is 'SWIFT Customer Security Programme (CSP)' certified and supports all Swift business areas and interfaces (FIN, Interact, RMA) on the one platform.

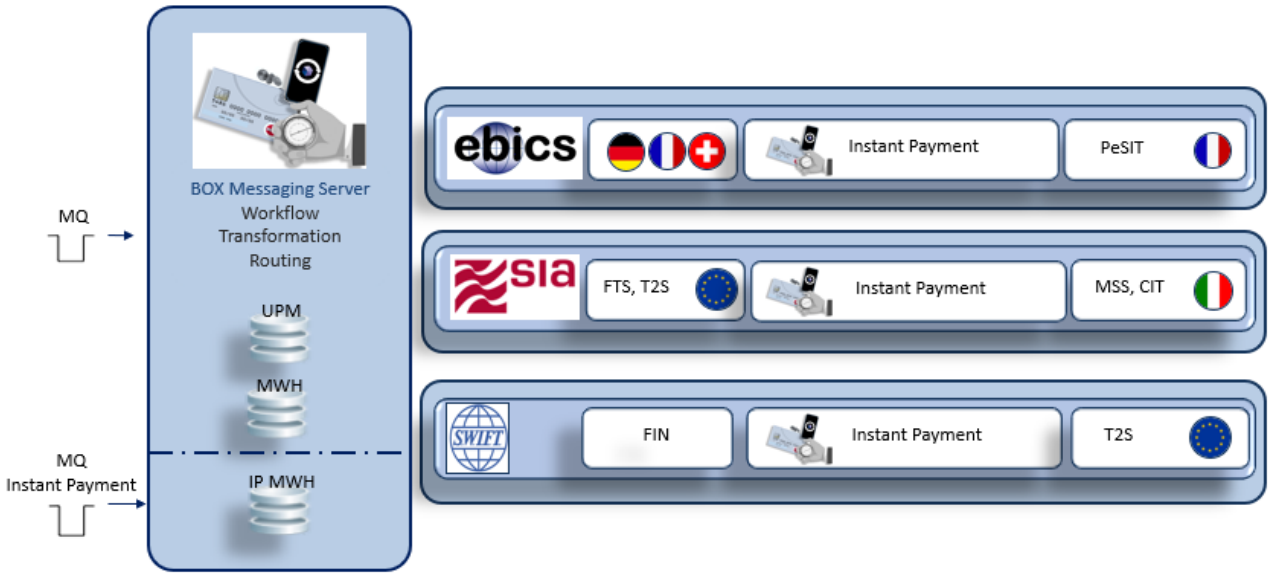


Figure 2 BOX Messaging Hub Messaging Overview

2.2 BOX and Instant Payments

BOX for Instant Payments is installed on the same code-based platform as it is used for other schemes, such as FileAct, FIN and MX. The setup for Instant Payments is based on messages being read from MQ, these messages are processed, transformed, enriched and finally written to a specific network gateway (SWIFT, SIA, EBICS).

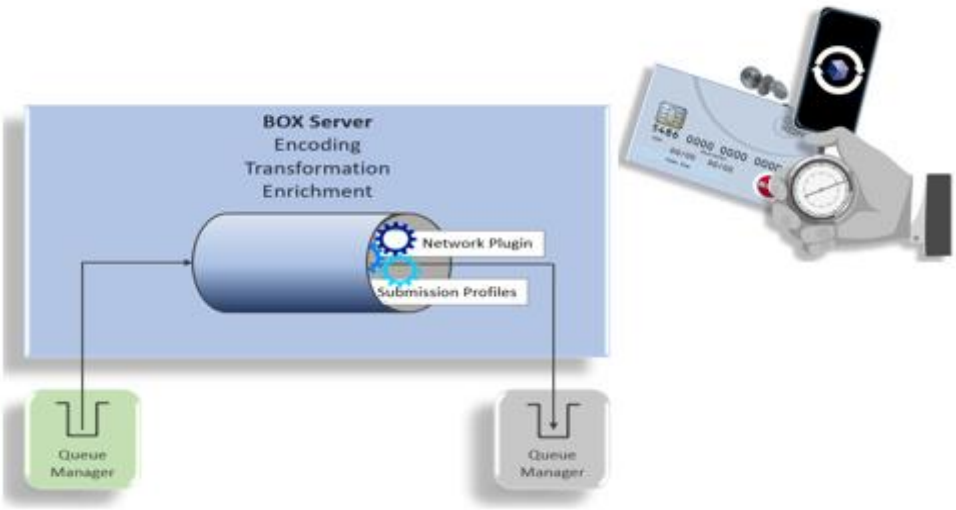


Figure 3 Instant Payments Message Processing

The recommended configuration for an Instant Payments System is two active sites, with an MQ and database cluster, with each site having an Active-Active BOX Cluster with access to the database and MQ cluster. Chapter 3 give an overview and the configuration of the architecture of available modes (persistence) in which BOX is supported to run.

2.3 Clearing and Settlement Mechanisms (Clearing Settlement Member, CSM)

BOX supports the following CSMs:

- RT1
- TIPS
- RT1/TIPS Instruction Party

2.4 Instant Payment Message Journal

The Message Journal dedicated to Instant Payment (IP) contains separate IP ISO20022 messages (payments) for a relatively short period of time (hours up until one day). An additional table for technical acknowledgments (Delivery Notifications) and Non-Repudiation Data (NR Data) is provided. GUI tasks reconcile Delivery Notifications and NR-Data. Also, IP ISO20022 messages are reconciled to form an SCTinst payment.

A shared IP database has been provided for containing the IP Message Journal, the SIA Instant Payment Endpoint and respective BX data.

2.5 Active-Active

The IP System can be set up as in Active-Active System

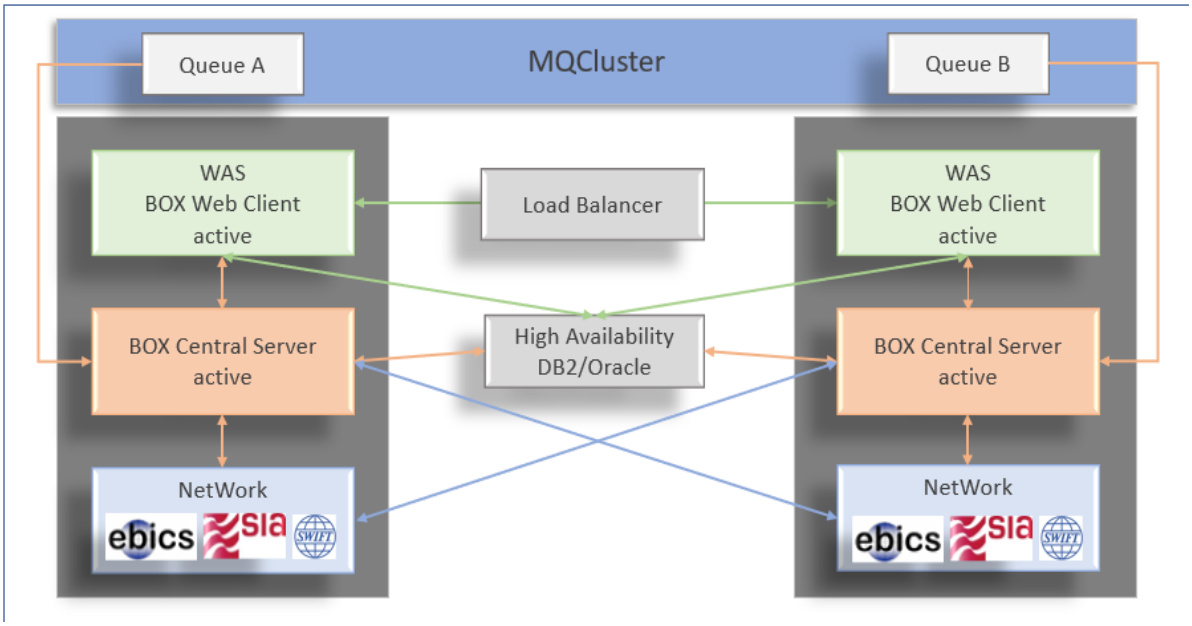


Figure 4 Active-Active Overview

2.5.1 Submission Profiles

Submission profiles are used to provide data enrichment with network envelope data. BOX has several submission profiles to handle all network related information. The following highlighted Submission Profiles are used for SWIFT and EBICS Instant Payments.

Available Profiles:

- Interact
- FileAct – Put File
- File Act – Get File
- SIA FLS – Send File
- SIA FTS – Send File
- SIA T2S – Send Message/File
- **EBICS – Send Instant Payments Realtime Message**
- EBICS – Interbank File Transfer
- **SWIFT – Send Instant Payments Realtime Message**

2.5.2 Example SWIFTNet Instant Payment Enrichment

Submission Profile SWIFT

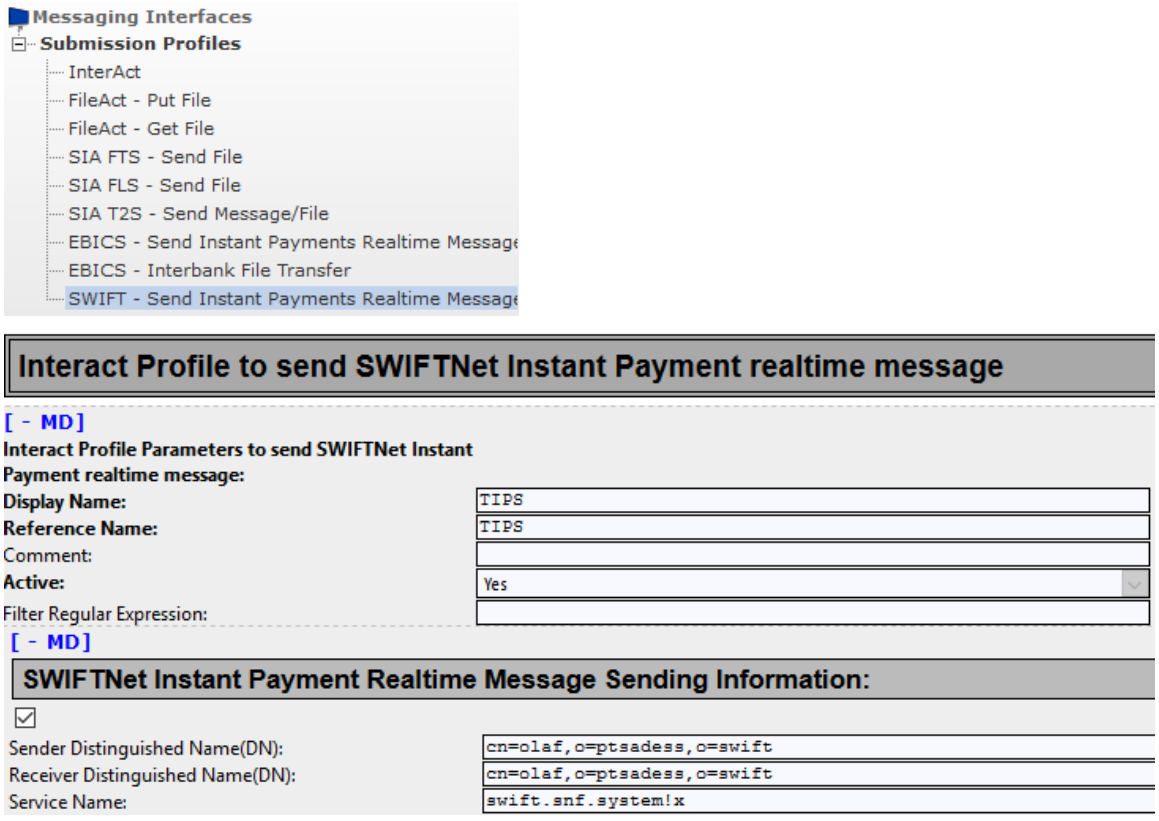
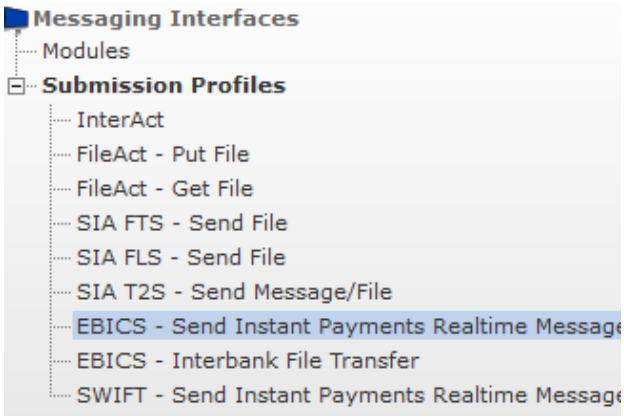


Figure 5 Submission Profiles – Example SWIFT Instant Payments

2.5.3 Example EBICS Instant Payment Enrichment

Submission Profile EBICS



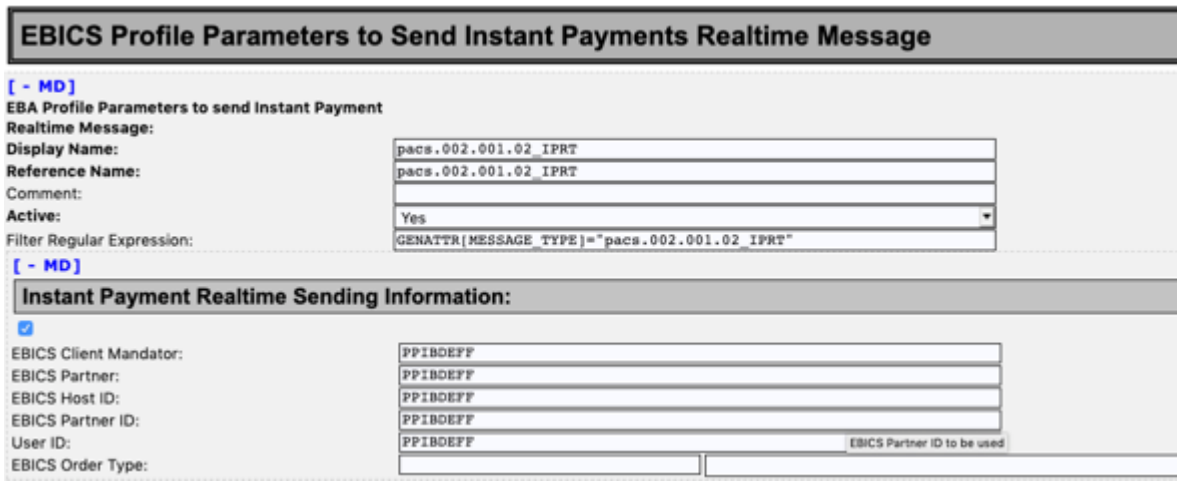


Figure 6 Submission Profiles – Example EBICS Instant Payments

2.6 Multi Network Connectivity

2.6.1 BOX connecting to EBICS TRAVIC

The following graph shows an Active-Active architecture, where two BOX instances use separate configuration stored in a static database and one instance Instant Payment Journal shared database. The connectivity is established via a configured preselected MQ channel. A specific configuration example can be found in chapter 4.1.2 and 5.2.

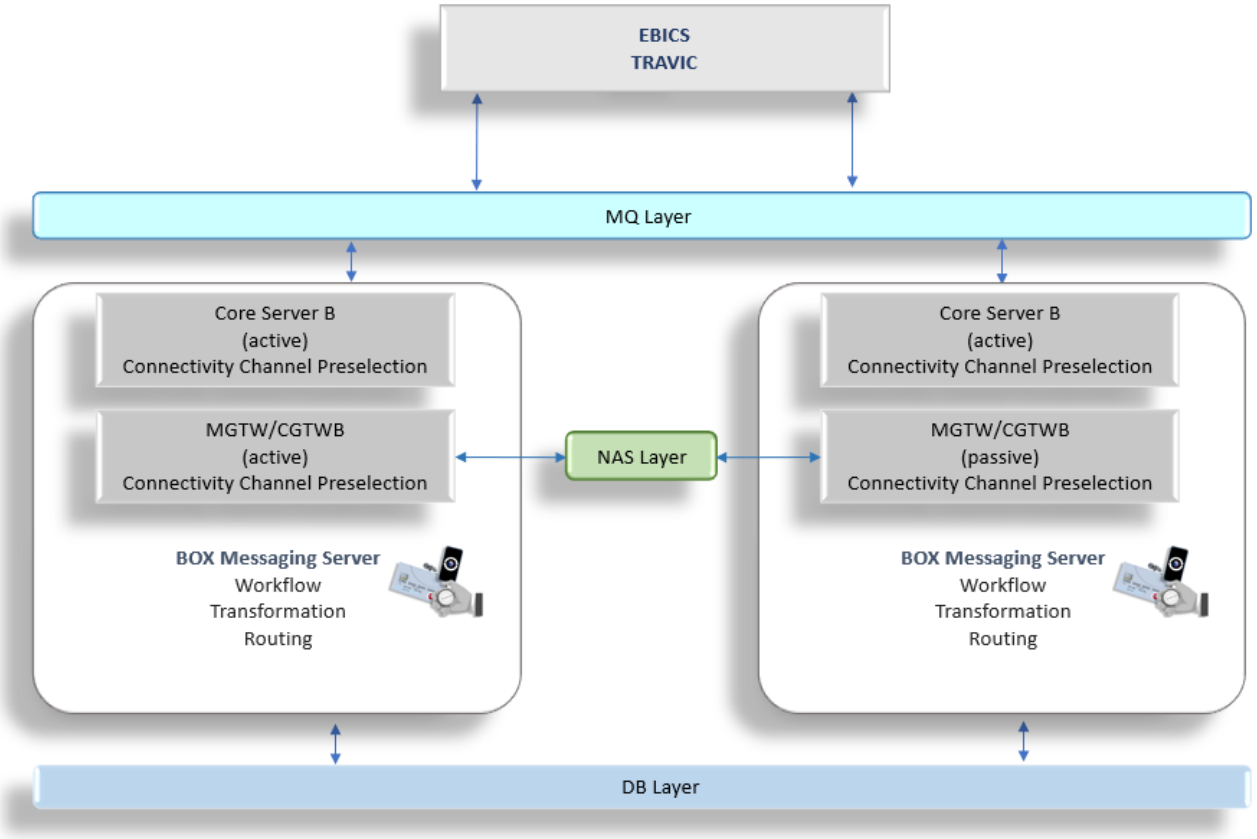


Figure 7 Exemplary EBICS Connection Overview

2.6.2 BOX connecting to SWIFT

BOX uses the Alliance Gateway Instant (AGI) Plugin to connect to the SWIFTNet Instant Messaging solution.

The Alliance Gateway Instant enables the exchange of ISO 20022 messages over IBM MQ or through the Alliance Messaging Hub Instant and acts as a local gateway between a customer's back-office application and the SWIFT network.

The following types of AGI setups are supported:

- a one-node AGI (a single host runs one AGI)
- a three-node AGI (the AGI software is installed on three hosts and operates as a single AGI over the three hosts)

Please refer to chapter 5.1.1 for a detailed description on the Alliance Gateway Instant (AGI) Plugin.

2.6.3 BOX connecting to SIA Net

BOX connects to the SIAnet via a Messaging Integration exchanging Real-Time messages and files.

BOX offers the following integrations with SIAnet

- SEPA / EBA CLEARING : Smart Integrator Advanced, File Transfer Service
- T2S : Smart Integrator Advanced, T2S Protocol
- RNI : EAS, Message Switching Service
- CIT (Assegni – Cheques) : EAS, Fast&Lite (File) Service

Extending into Instant Payments

- SCT-Inst : TIPS, RT1

Figure 8 BOX Messaging Hub Connecting to SIAnet

Real-Time message and bulk message exchange are done using the SIA Smart Integrator Advanced. BOX registers its infrastructure to SIAnet Central Services and is then connected via a secure link to SIAnet and the transport configuration (queue manager, queue names, queue options) for the link to communicate with the SIAnet infrastructure.

By registration BOX, as a Business User, joins a Domain (DOM) with a Business User Address. BOX represents a Business User.

For Instant Payments, the Instant Message eXtended (IMX) Service provided by the FEMS XS is used. The following graph depicts the phases Command Phases, including the Logon and Subscription and the Business Phases Send and Receive.

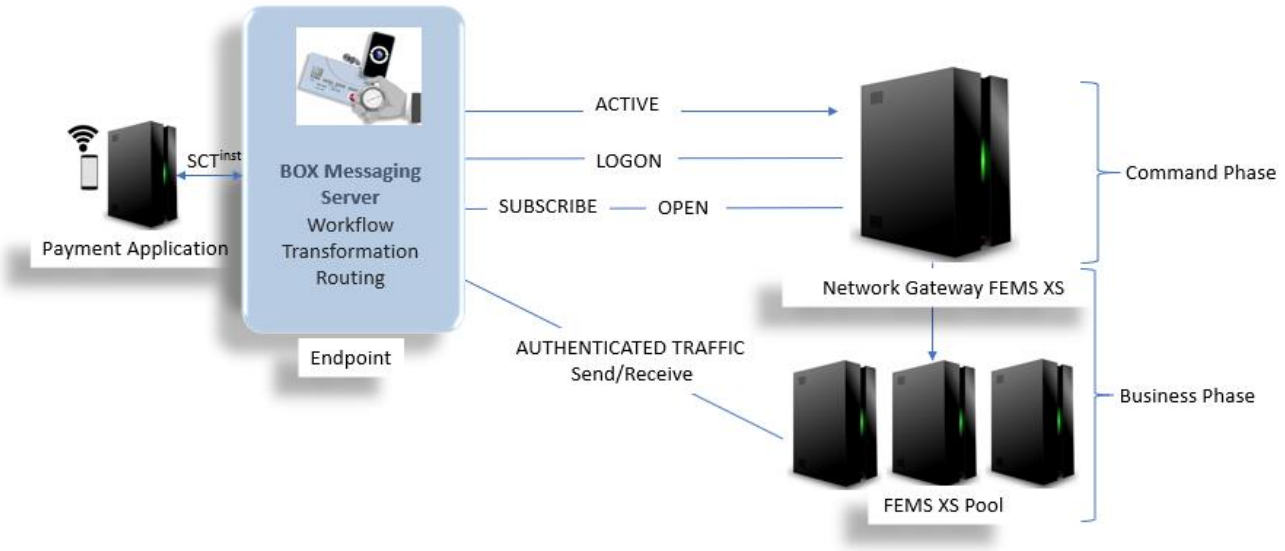


Figure 9 Connecting to SIAnet: Command- and Business Phases

3 Persistence Modes and their Configuration

IMPORTANT

Never mix Instant Payments with non-Instant Payments installations within 1 BOX (defined through (logical) databases)

Use only configuration options mentioned before. No deviations without consulting Intercope! Never use untested configurations!

BOX has different persistence modes implemented to serve the option to grade the level of persistence. These different modes will be explained throughout the following sub-chapters.

Available Modes

Value, String	Explanation
`InstPmtNoPersistence`	Instant Payment MPS using no persistence • Notification reconciliation data, Notifications create own MPS • Technical Acks • Can never change persistence level and should use only appropriate LCGs
`InstPmtJrnPersistence`	Instant Payment MPS using InstPmt journal persistence • Notification reconciliation using Cache resp. InstPmt Msg Journal • MPS-ID allocation resulting in IP_JRN_SEQNO sequence • Instant Payment created MPS may also use InstPmtMPSPersistence • Single Queue-Manager per BOX-Flow! • InstPmtJrnPersistence created MPS may raise persistence level to IP_MPS when being halted or stopped in an Application Queue.
`InstPmtMPSPersistence`	Instant Payment MPS using InstPmt journal and full MPS persistence • Traditional BOX transaction handling • Attach delivery reports (also) to MPS • MPS-ID allocation resulting in IP_JRN_SEQNO sequence • Instant Payment created MPS may also use InstPmtJrnPersistence
`FullPersistence`	Instant Payment MPS using full persistence and detailed transaction model • Traditional BOX transaction handling • Attach delivery reports (also) to MPS

3.1 Operating BOX in Non-Persistence Mode (InstPmtNoPersistence)

The non-persistence mode allows BOX to operate without database connectivity. Instant Payment MPS are not written to a (any) database, using non-DB ProcessingSequenceID allocation, a change of persistence level is not possible. The Transaction handling is attached to MQ.

The initial database connection is used to read the workflow configuration, but no operational message data is stored in the database. To allow this mode, not only the processing of messages, which are not written to a database with all its attributes, but also the composition of messages to be delivered has changed. It is important to understand, the concept of message processing to avoid any configuration mistakes.

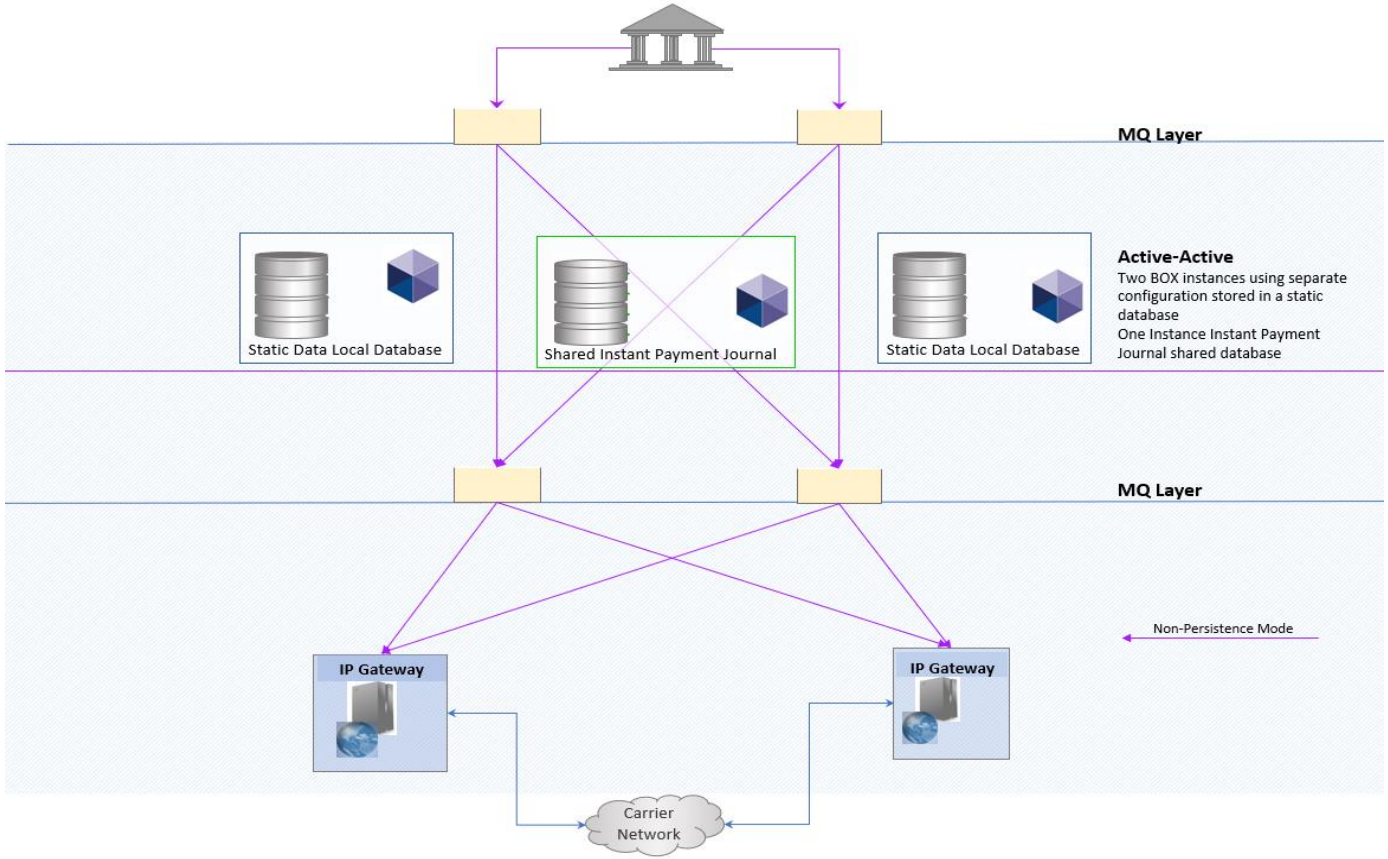


Figure 10 Non-Persistence Mode - Overview

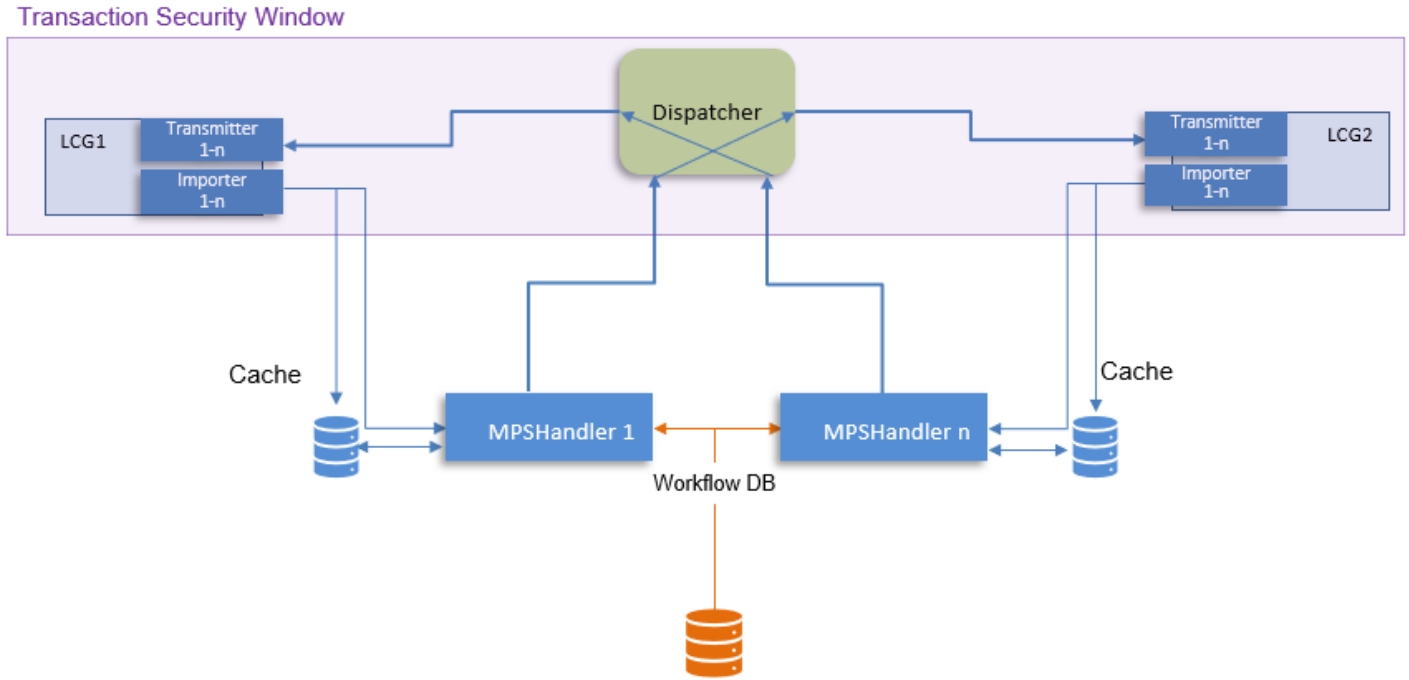


Figure 11 Non-Persistence Mode - Configuration

Dispatcher

Collecting all data for delivery and submits it to the destination LCG Transmitter.

LCG (Local Channel Group)

The Transmitter sends the order received from the Dispatcher to the destination. The Importer receives message and writes it to the Cache. Unit of Work is the Outgoing transaction, unless rolled back.

MPS Handler

Processes the configured Workflow (DLI, TGI, CPI, SBI, WTI).

3.1.1 Message Processing

The Instant Payments message flow can be roughly devided into 4 important steps:

- STEP 1: Message is read from a queue und transferred to the BOX Server for processing
- STEP2: Message is processed involving a transformation, an enrichment and encoding
- STEP3: Message is written to the SWIFTnet network, a technical response/an acknowledgement is received
- STEP4: An acknowledgement is transferred to the sender. If the delivery was successful, message is now deleted from the queue. If the delivery was unsuccessful, the cycle of STEP 1 till STEP 4 will restart, until the configured 'Backout Count' has reached its configured maximum. Please refer to chapter 3.1.3.

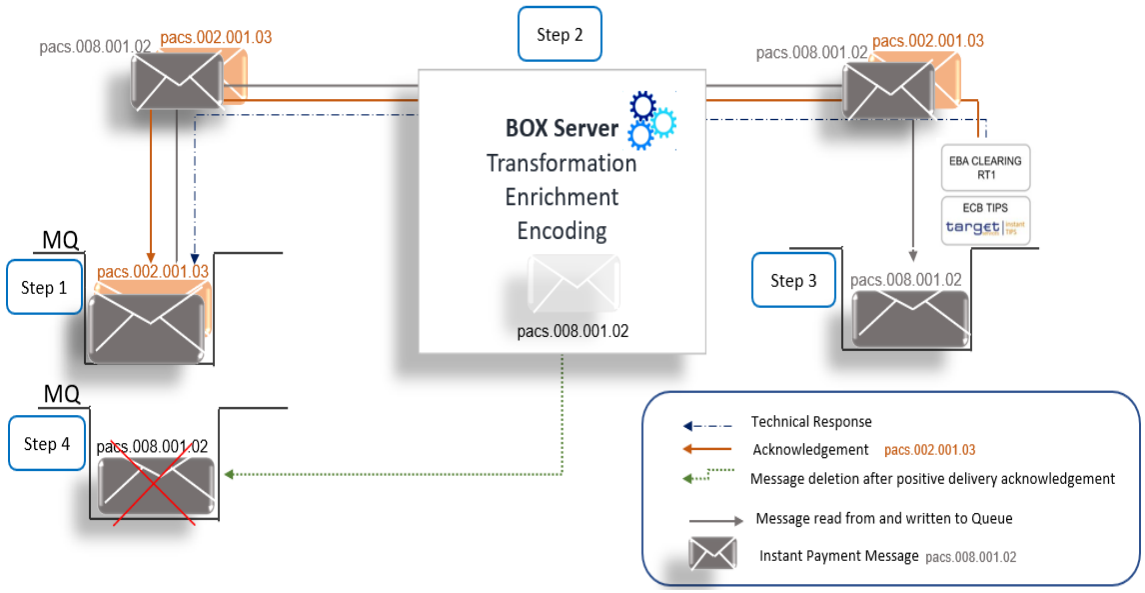


Figure 12 Overview Instant Payment Message Processing

3.1.2 Address Book Cache

Address book caching has been implemented to accelerate message processing by avoiding DB-access to address books during message dispatching. In such scenarios routing destinations are preferably configured inside a pattern through delivery instruction pattern destinations in a fixed manner as recipients or (even faster) recipient addresses.

Example

```
[MPS_HANDLER]
   CACHE_AB01          node:demofin:demofin:ISP_AddressBook
```

Parameter	Description
CACHE_AB<postfix>	This parameter may be used to accelerate message processing by avoiding DB-access to address books during message dispatching. In such scenarios routing destinations are preferably configured inside a pattern through delivery instruction pattern destinations in a fixed manner as recipients or (even faster) recipient addresses. Address book specification (keyword value) uses following syntax: <OwnerType>:<ClientPrefix>:<OwnerShortname>:<AddressbookShortname> with OwnerType = Node | User, ClientPrefix is the client prefix of the owner of address book, Shortname is UPM-short name of owner of address book and AddressbookShortname specifies the short name as assigned to this address book. Actual keywords may be CACHE_AB01 or CACHE_AB_FIN, it is possible to cache several address books. Be aware that (currently) changes on cached address books might refresh the cache only after server module restart.

Table 3.1-I Parameter CACHE_AB

This parameter might be used in persistent message processing also. Fully non-persistent messages (absolutely no database entry) use a different MPS-ID allocation algorithm.

IMPORTANT
As configuration data is read from the database, MP/O modules still do require database access during start-up. Server modules processing non-persistent messages should disable Gap-Detection (other security measure might be enabled).

3.1.3 Backout Count

The Backout Count is a vital part of the Instant Payments Message Processing in Non-Persistence Mode, as this configuration limits the number of processing cycles, hence preventing the BOX Server falling into a processing loop - in the event of a consecutive message delivery failure.

Parameter	Default	Description
SECTION Exchange Adapter F002		
TRASH_BACKOUT_LEVEL	0	This parameter may be used to trigger a special MQ exception handling for MQ queue entries using an excessive backout count. Setting this parameter to 0 disables this function. This special handling includes copying the message into configured TRASH-Queue (parameter TRASH_QUEUE_NAME in this section), requested MQ-Reporting and copying into dead-letter-queue (if configured: USE_DEAD_LETTER_QUEUE). If fully persistent processing is configured, then an exception MPS is created also.

Parameter	Default	Description
SECTION Exchange Adapter F002		
		If a queue entry carrying a backout level equal to or larger than the configured value is read, then the described processing is performed. See also parameter EXCEPTION_BACKOUT_LEVEL in this same section and be aware that this check is performed prior to exception-backout check. This parameter is mandatory for MQ transports applied in LCGs which create non-persistent MPS (see keyword ([LCG<lcgname>.PEXA].MPS_PERSISTENCE_LEVEL)
EXCEPTION_BACKOUT_LEVEL	0	This parameter may be used to trigger a special exception workflow processing for MQ queue entries using an excessive backout count. Setting this parameter to 0 disables special processing. If a queue entry carrying a backout level equal to or larger than the configured value is read then an Exception MPS is created and the LCG exception pattern (e. g. [LCG<name>.PEXA].DEFAULT_EXCEPTION_SHORTLABEL) is used to start the workflow for this message. See also parameter TRASH_BACKOUT_LEVEL in this same section. This parameter is mandatory for MQ transports applied in LCGs which create non-persistent MPS (see keyword ([LCG<lcgname>.PEXA].MPS_PERSISTENCE_LEVEL) !

Table 3.1-II Parameter Backout Count

Please refer to the Appendix, as the backout count has been added to certain Reception Reports.

3.1.4 Delivery Composition

The delivery composition describes the parts, a message is made of for delivery and can be configured according to specific requirements. The configuration can be done either in the respective channel or in the Delivery Instruction of the workflow, though it is quite often simply set to the channel's settings as default.

The general delivery composition and its possible values to be configured in the channel's configuration:

Parameter	Default
SECTION [LCGXXX]	
DEFAULT_DELIVERY_COMPOSITION	(derived from channel type)
Description	
0x0000100: include rendered content 0x0000010: include original content (mutual exclusive with 0x0000100) 0x0000001: include report data 0x0001000: include report item (only possible with delivery reports) 0x0002000: include generic attributes of content version 0x0010000: include effective properties of default owner	

Table 3.1-III DEFAULT_DELIVERY_COMPOSITION = 0x0010101

IMPORTANT

As messages are not written to a database in the Non-Persistence Mode, the delivery composition must change.

The Non-Persistence delivery composition and its possible values to be configured in the channel's configuration or workflow 'Delivery Instruction':

Parameter	Default
SECTION [LCGXXX]	
DEFAULT_DELIVERY_COMPOSITION	0x0000010 (derived from channel type)
Description	
0x0000010: include original content	

Table 3.1-IV `DEFAULT_DELIVERY_COMPOSITION = 0x0000010`

3.2 Operating BOX in Instant Payment Journal Persistence (InstPmtJrnPersistence)

The ‘Instant Payment Journal Persistence’ mode refers to Instant Payment messages being written to a shared Instant Payment Database using the Instant Payment-Database ProcessingSequenceID allocation. It is possible to enhance the persistence level to level MP_MPS_PERSISTENCE_IP_FULL. The transaction handling is attached to MQ.

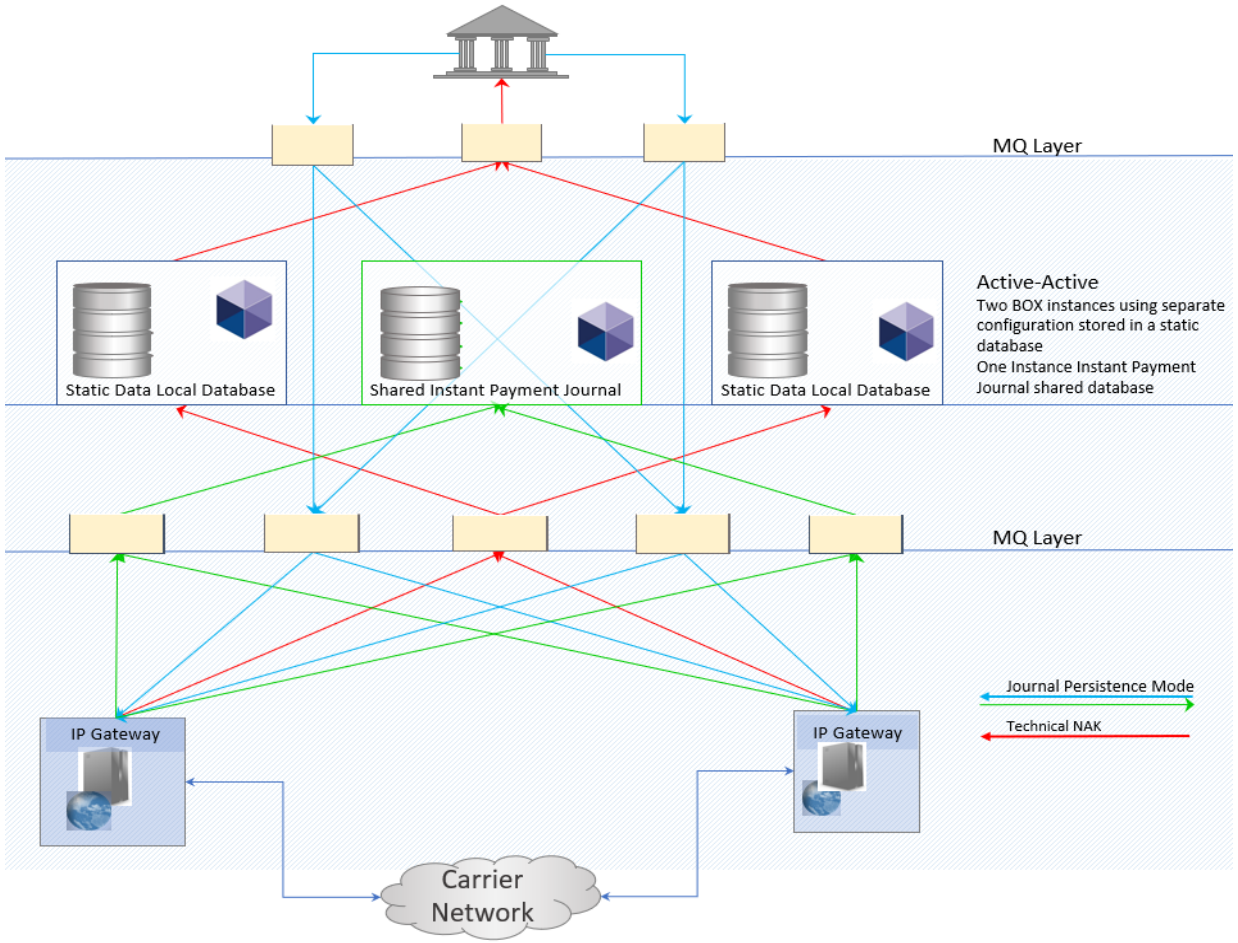


Figure 13 Instance Payment Journal Persistence Mode - Overview

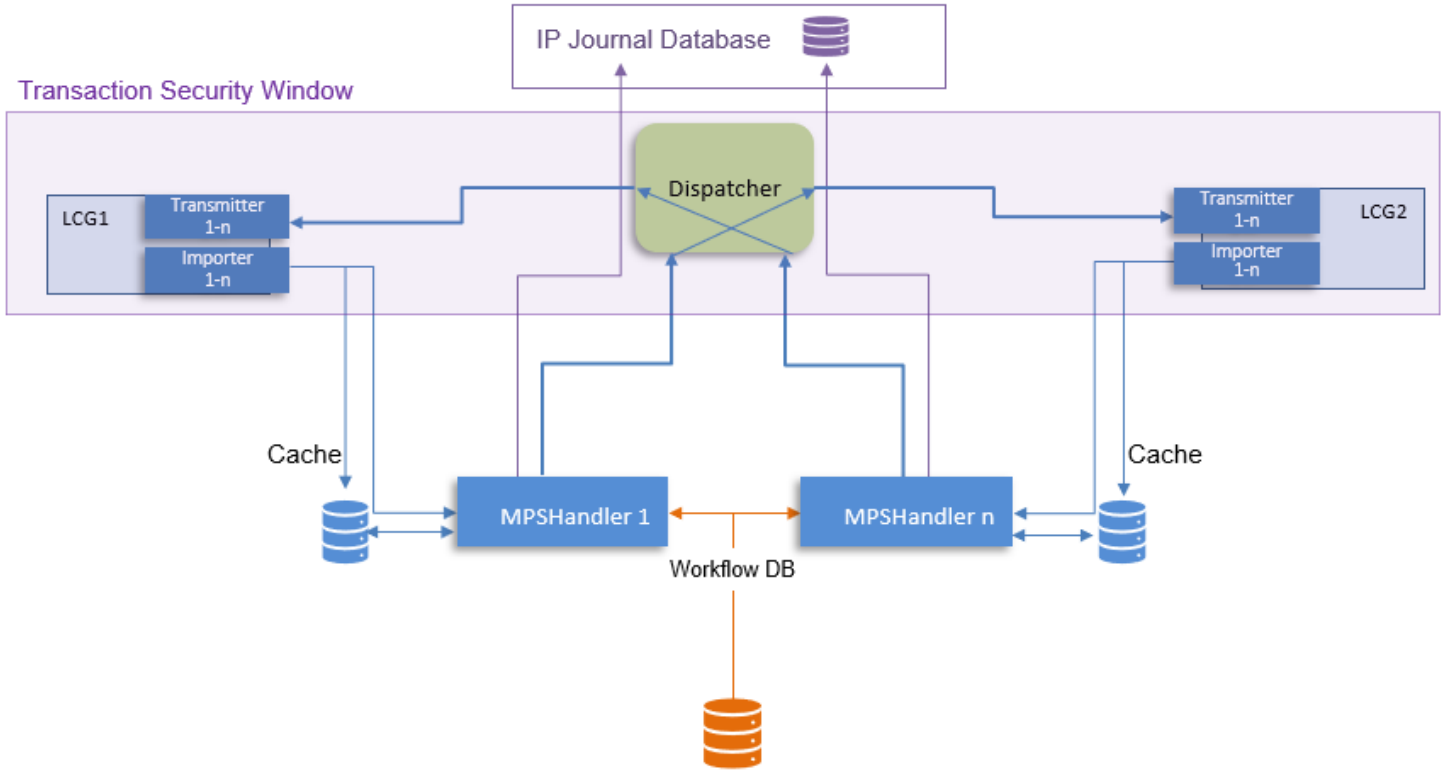


Figure 14 Instance Payment Journal Persistence Mode – Configuration

3.3 Operating BOX in Message Processing Sequence (MPS) Full Persistence (InstPmtMPSPersistence)

Instant Payment MPS is written to the Instant Payment - Journal in a shared IP-database and written to MPS tables in the local database, using the Instant Payment-Database ProcessingSequenceID allocation traditional transaction handling based on MPS instruction persistence in the database.

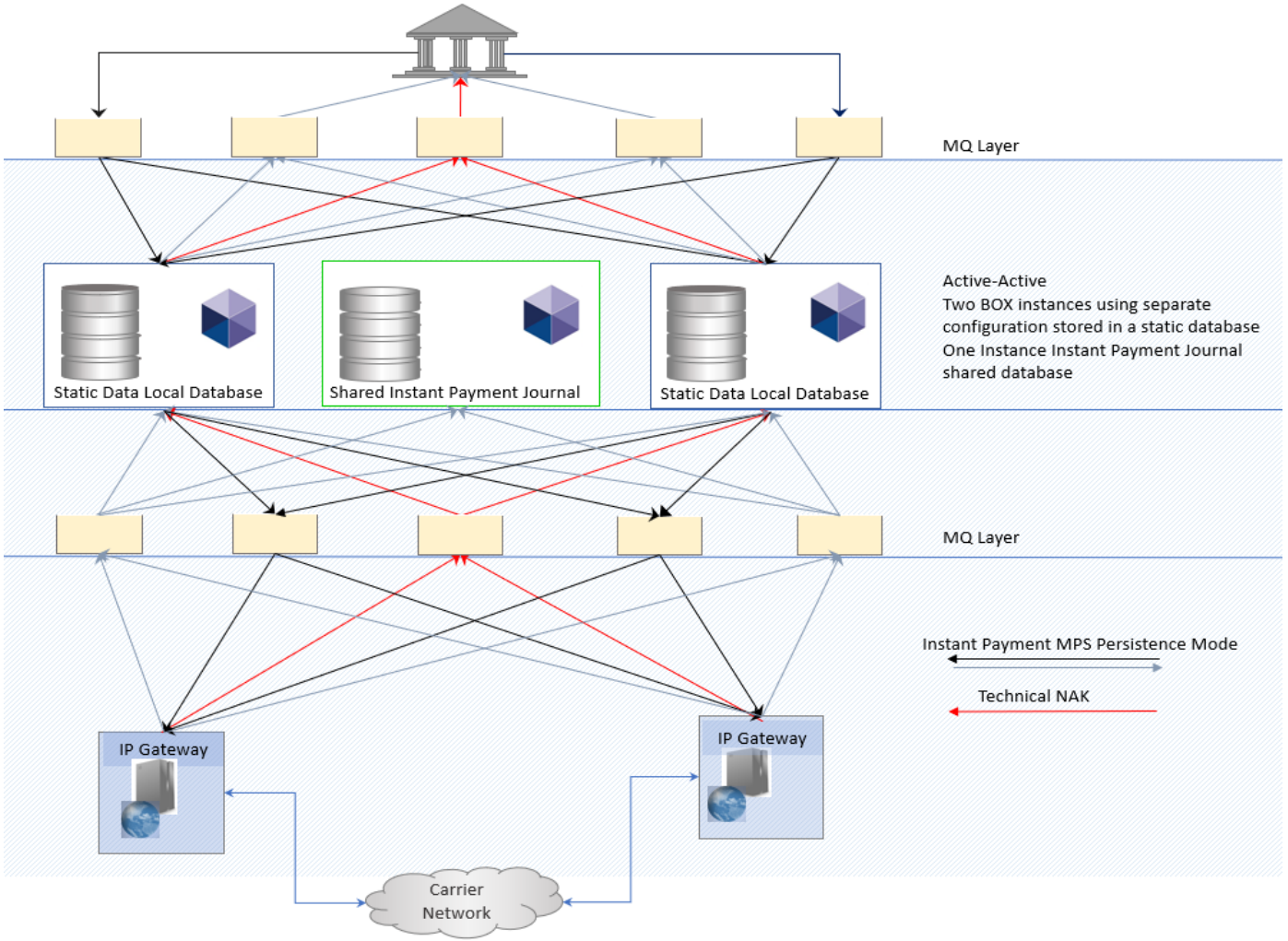


Figure 15 Instant Payment MPS Persistence Mode - Overview

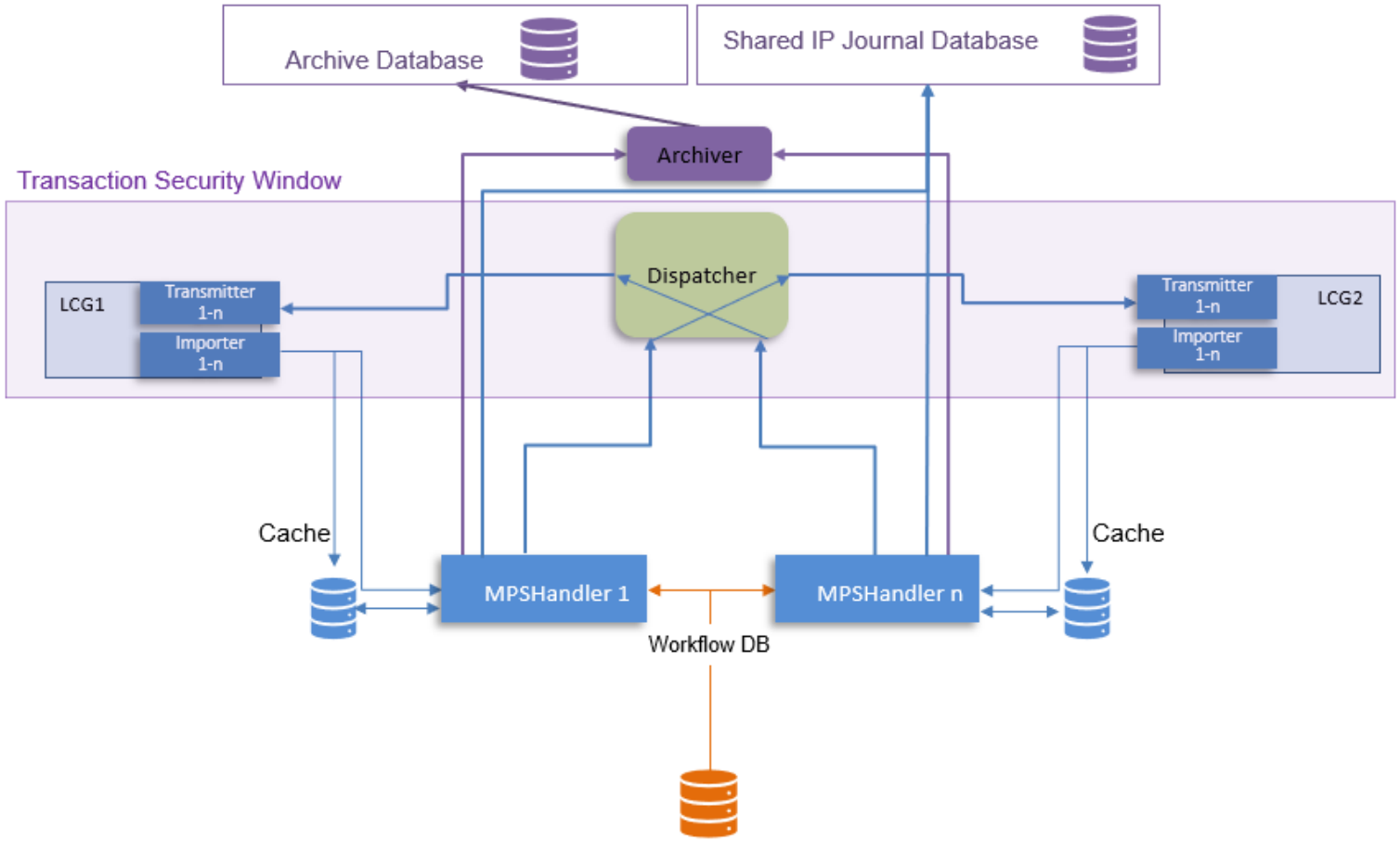


Figure 16 Instant Payment MPS Persistence Mode - Configuration

3.4 Operating BOX in Full Persistence Mode

The Instant Payment MPS' are written to MPS tables in the local database, using the traditional transaction handling based on MPS instruction persistence in the database.

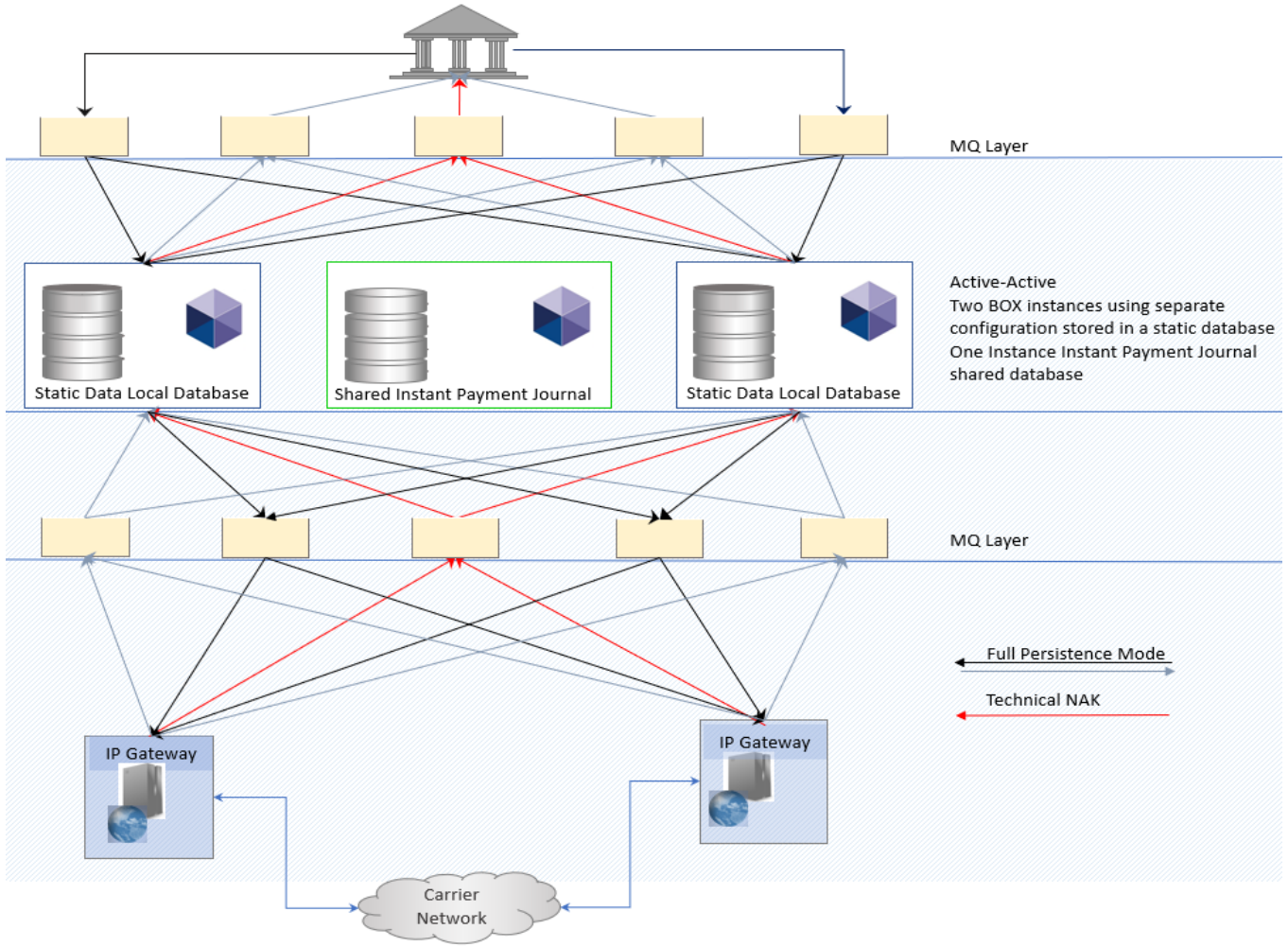


Figure 17 Full Persistence Mode - Overview

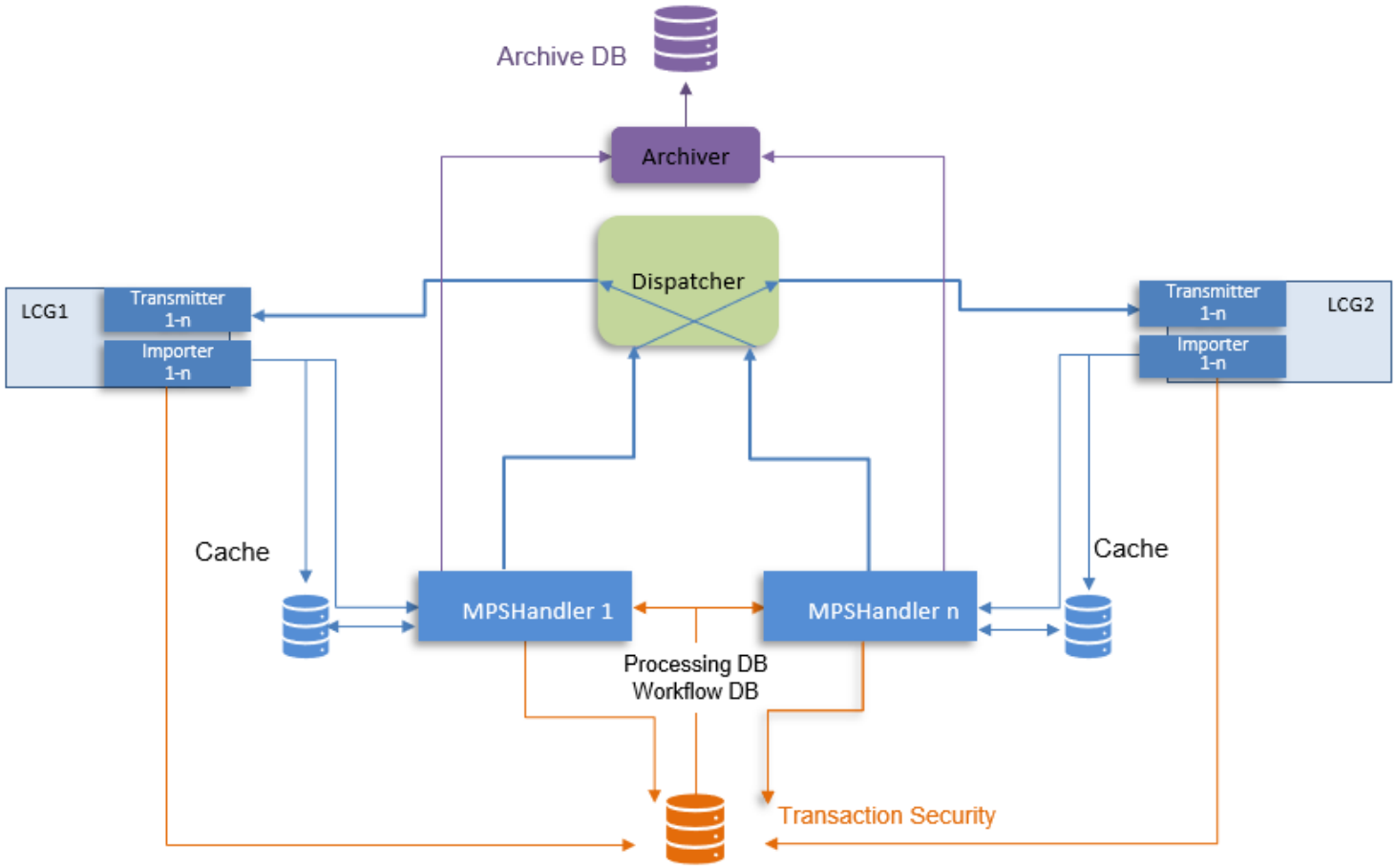


Figure 18 Full Persistence Mode - Configuration

Dispatcher

Collecting all data for Delivery and submits it to the destination LCG Transmitter.

Archiver

Archives the MPS to the Online Archive, when processing is finished.

LCG (Local Channel Group)

Transmitter sends the Order received from the Dispatcher to the destination Importer, which in turn receives a message and writes it to the Cache and the Database (Unit of Work)

MPS Handler

Processes the configured Workflow (DLI, TGI, CPI, SBI, WTI)

3.4.1 Duplicate Check

Duplicate checking, if required, is only possible in Full Persistence Mode.

3.4.2 Error Handling

In case of technical errors, a NAK results in the generation of an MPS, which is written to a dedicated error queue and consequently dealt with according to the predetermined error handling on customer side.

3.4.3 BOX Archive for Instant Payments

3.4.3.1 Full Persistence Mode (FullPersistence)

The Full Persistence Mode caters for a possible roll back and is based on caching message data before writing it to the archive database.

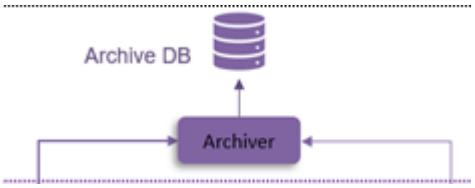


Figure 19 Archive Database Tables

Dispatcher

The Dispatcher collects all data for delivery and submits it to the destination LCG Transmitter.

Archiver

This archives the MPS to the Online Archive, when processing is finished.

LCG (Local Channel Group)

The transmitter sends the order received from the Dispatcher to the destination. The importer receives message and writes it to the Cache. The Unit of Work is the Outgoing transaction, unless rolled back.

3.4.3.2 Configuration of BOX Archiver

Keeping track of messages for Instant Payments in a Non-Persistence Mode is quite different to a Mode, where messages are written to a database and the archive holds here a central role as a journal and an archive rather than just an archive.

Viewing messages is a central part of the BOX Web Client, but without an existent Message Warehouse, views must be directed toward the archive. But within this context, it is by choice to operate an archive, it is not mandatory for the Non-Persistence Mode.

The configuration is set within the respective channel configuration and is no different to any other archive setup and

The Archiver can be configured either to run as an external tool ('offline') or to run within the Server ('online').

3.4.3.3 Archiver within the Server

If the Archiver runs within the Server, the Server's configuration file must include the section `[SECURITY]` containing the security related parameters and the section `[BOX_ARCHIVER]` containing at least the mandatory parameter `SUPPORTED_MESSAGE_TYPES`.
Additionally, the parameter `ARCHIVER` in section `[MPS_CACHE]` must specify the archive library `box_archiver`.
The database connection related parameters are per default read from the section [`DB_INTERFACE]` in the Server's configuration file (i.e. the Archiver uses the same database connection as BOX).

Section	Default	Description
`Section [BOX_ARCHIVER]`		
`SUPPORTED_MESSAGE_TYPES`		This parameter defines a list of message types that are archived by the BOX Archiver. Currently the following message types are allowed: `FIN, SWIFT (alias for FIN), MX, FACT, SIA_FTS, RNI, T2SMSG, T2SFILE, SIA_FLS`
`IGNORED_MESSAGE_TYPES`		This parameter defines a comma separated list of message types that are to be ignored by the BOX archiver. The message types that can be used are the same as for `SUPPORTED_MESSAGE_TYPES`. If the message type of an MPS is not listed in `SUPPORTED_MESSAGE_TYPES` but in `IGNORED_MESSAGE_TYPES`, the archive flag of the MPS will not be set to `NEVERARCHIVED` but to `NOTARCHIVED`.
`DATA_COMPRESSION_METHOD`		Data compression method used for archiving; Default value is 'No compression'.
`ARCHIVE_FILE_DATA`	`NO`	The box_archiver allows archiving file transfer data for SWIFT FileAct, T2S File and SIA FTS messages. If this config parameter is set to YES (default value is NO), the file transfer data will be written to a separate table (<qualifier>.FACTARCHFD_<partition_postfix>[<year_postfix>]).
`Section [MPS_CACHE]`		
`CACHE_SIZE`		1000
`MAX_ITEM_SIZE`		100
`ARCHIVER`	`box_archiver`	This parameter specifies an application (and customer) specific archive library. This archive library is used to generate an archive entry for an MPS. An empty value disables archiving

Table 3.4-V Database Connection Parameter

Further parameters in these sections and their description can be found in the **BOX Configuration Guide**.

3.4.3.4 ARCHIVER as External Tool

If the Archiver runs as an external tool, the configuration file of the tool must contain the following sections:

`[SECURITY]`
This section is mandatory. It contains security related parameters.

Section	Default	Description
Section [BOX_ARCHIVER]		
SUPPORTED_MESSAGE_TYPES		This parameter defines a list of message types that are archived by the BOX Archiver. Currently the following message types are allowed: FIN, SWIFT (alias for FIN), MX, FACT, SIA_FTS, RNI, T2SMSG, T2SFILE, SIA_FLS
IGNORED_MESSAGE_TYPES		This parameter defines a comma separated list of message types that are to be ignored by the BOX archiver. The message types that can be used are the same as for SUPPORTED_MESSAGE_TYPES. If the message type of an MPS is not listed in SUPPORTED_MESSAGE_TYPES but in IGNORED_MESSAGE_TYPES, the archive flag of the MPS will not be set to NEVERARCHIVED but to NOTARCHIVED.
DATA_COMPRESSION_METHOD		Data compression method used for archiving; Default value is 'No compression'.
ARCHIVE_FILE_DATA	NO	The box_archiver allows archiving file transfer data for SWIFT FileAct, T2S File and SIA FTS messages. If this config parameter is set to YES (default value is NO), the file transfer data will be written to a separate table (<qualifier>.FACTARCHFD_<partition_postfix> [<year_postfix>]).
Section [DB_INTERFACE]		
DATABASE_NAME		Name of database used by BOX.
DATABASE_QUALIFIER		Name of database qualifier used by BOX.
DATABASE_USERNAME		Name of database user used for logging in
DATABASE_PASSWORD		Encrypted password for logging into database
DATA_SOURCE_NAME		Leave blank!
DB_INTERFACE_LIBRARY		mp_db2cli (for db2) or mp_oraoci (for Oracle).
Section [MPS_CACHE]		
CACHE_SIZE		1000
MAX_ITEM_SIZE		100
ARCHIVER	box_archiver	This parameter specifies an application (and customer) specific archive library. This archive library is used to generate an archive entry for an MPS. An empty value disables archiving

Table 3.4-VI External Archiver Configuration Parameters

Example:

```
[SECURITY]
   SECURITY_ZIP_FILE              security.zip
   BASE_SECRET_LIST               ;

[BOX_ARCHIVER]
   SUPPORTED_MESSAGE_TYPES        FIN, FACT; MX
```

```
   DATA_COMPRESSION_METHOD          GZIP

[MPS_CACHE]
   CACHE_SIZE                       120
   CACHE_CONTENT                    YES
   MAX_ITEM_SIZE                    200
   ARCHIVER                         box_archiver

[DB_INTERFACE]
   DATA_SOURCE_NAME                 ; Leave blank
   DATABASE_NAME                    ; to be defined
   DATABASE_QUALIFIER               ; to be defined
   DATABASE_USERNAME                ; to be defined
   DATABASE_PASSWORD                ;
   DB_INTERFACE_LIBRARY             mp_db2cli ;MAX_CONNECTIONS       30; make sure
the value is high enough.; Especially when the offline
archiver tool; is used, the value must be at least as
high; as the number of threads to be started by; the
archiver tool (start parameter (-t)).
```

4 Configuration

4.1 Exemplary Instant Payment SWIFT Configuration

The Instant Payment Server configuration:

```
DATABASE
  [INSTPMT_DB_INTERFACE]
    CONNECTION_TIMEOUT = 10
    CONNECTION_TIMEOUT_CHECK_INTERVAL = 60
    DATABASE_NAME = $$R$DB_DATABASE_IP_NAME$
    DATABASE_PASSWORD = $$R$DB_DATABASE_IP_ENC_PWD$
    DATABASE_QUALIFIER = $$R$DB_DATABASE_IP_QUALIFIER$
    DATABASE_USERNAME = $$R$DB_DATABASE_IP_USER$
    DATA_SOURCE_NAME = $$R$DB_DATABASE_IP_NAME$
    DB_INTERFACE_LIBRARY = mp_oraoci_12c
    EXECUTION_TRACE = 3
    MAX_CONNECTIONS = 30

DELETER
  [INSTPMT_DELETER]
    DELETE_CANCELLED_MPS = YES
    DELETE_PENDING_JRNL = NO
    DELETE_PROCESSED_MPS = YES
    FRIDAY_INTERVAL = J02:00-06:00
    JRNL_LIST_SIZE = 100
    MAX_DELETEJRNL = 10
    MAX_DELETEMPS = 500
    MONDAY_INTERVAL = J02:00-06:00
    RESPECT_ARCHIVE_FLAG = NO
    RESPECT_EXPIRATION_DATE = YES
    RESPECT_EXPIRATION_TIME = YES
    SATURDAY_INTERVAL = J02:00-06:00
    SUNDAY_INTERVAL = J02:00-06:00
    THURSDAY_INTERVAL = J02:00-06:00
    TUESDAY_INTERVAL = J02:00-06:00
    WEDNESDAY_INTERVAL = J02:00-06:00
    WORKER_COUNT = 0

JOURNAL
  [IP_JOURNAL_WRITER]
    DATA_COMPRESSION_METHOD = NONE
    EXPIRATION_TIME

LCG
```

```
[LCG<IPSWIFT>]
    2PC_FOR_MESSAGES = NO
    2PC_FOR_REPORTINGS = NO
    APPLICATION_GROUP_NAME = IPSWIFT
    CACHE_CONTENT = YES
    CACHE_NO = 779
    CACHE_SIZE = 200
    CGTW_HOST
    CHANNEL_TYPE = IP-RT
    DISABLE_LCG = NO
    IMPORTER_COUNT = 3
    IP_JRN_WRITER = ip_journal_writer
    LCG_OWNER = IP:BANKINST
    MAX_ITEM_SIZE = 1000
    OVERFLOW_CACHE_SIZE = 0
    TRANSMITTER_BLOCKTHRESHOLD = 0
    TRANSMITTER_COUNT = 1
    TRANSMITTER_UNBLOCKTHRESHOLD = 0
    TRANSPORT_COST = 0
[LCG<IPSWIFT>.F002]
    ADDITIONAL_INBOUND_QUEUE_LIST = PART.TQ, TIPS.RQ
    DEFAULT_OUTBOUND_QUEUE = PART.EQ
    DEFAULT_OUTBOUND_QUEUE_MANAGER = MQTEST
    DEFAULT_REPLY_QUEUE = PART.TQ
    DEFAULT_REPLY_QUEUE_MANAGER = MQTEST
    DELIVERY_REPORT_GENERATION = DLV_REPORT_GEN_IMMEDIATE
    EXCEPTION_BACKOUT_LEVEL = 2
    GENERATE_COMMAND_REPORT = NO
    INBOUND_QUEUE = PART.RQ
    LOCAL_QUEUE_MANAGER = MQTEST
    MESSAGE_DUMP_LIMIT = 100000
    MQ_USER_IDENTIFIER
    PLUGIN_LIBRARY_NAME = expgi_swift_agi
    SERVER_RESPONSE_MATCHING = off
    TRASH_BACKOUT_LEVEL = 2
    TRASH_QUEUE_NAME = PART.TQ
[LCG<IPSWIFT>.F002.SWIFT_AGI_PLUGIN]
    LAU_KEY_1 = XXXXXXXXXXX
    LAU_KEY_1_ID = A
    NOTIFICATION_REQUIRED = ALWAYS
    SUPPRESS_LAU_VERIFICATION = NO
[LCG<IPSWIFT>.PEXA]
```

```
CREATOR_PREFIX = IP
DEFAULT_CREATOR = BANKINST
DEFAULT_EXCEPTION_SHORTLABEL = INSTP_XXX_SWI_014_Exception_Messages
DEFAULT_IPS_SHORTLABEL = INSTP_XXX_SWO_013_Process_Incoming_Message
DEFAULT_MPS_INITMODE = 2
DEFAULT_REPORTING_SHORTLABEL
DELIVERY_MONITOR = YES
DEVICE_TYPE = 0xF002
IMPORT_CHECK_CYCLE = 5
MONITOR_CARRIER_DELIVERY = NO
MPS_PERSISTENCE_LEVEL = InstPmtMPSPersistence
PEXA_LIBRARY = eximf002_cl
STORAGE_PERIOD
```

4.1.1 MPS-Cache and LCG Definitions

Please note, the LCG owner needs to be set (to company), since the Instant Payment Messaging Journal is company specific.

Special Attention must be payed to the MPS-Cache, if using IP Journal Persistence and a filtered response reader is used for Notifications (SIAnet only), then both LCGs involved in the MPS creation and submission need to use the same MPS-cache!

```
[LCG<TEST1MQ>]
  ALERT_ON_EXCEPTION_MPS          YES
  APPLICATION_GROUP_NAME          TESTMQ
  CACHE_CONTENT                   YES
  CACHE_NO                        777
  CACHE_SIZE                      150
  CGTW_HOST                       ; P:2,1; <Protocol>:ModuleID, LCG-Number
  CHANNEL_TYPE                    MQ
  IMPORTER_COUNT                  5
  IP_JRN_WRITER                   ip_journal_writer; put lib here
  LCG_OWNER                       demofin:Bank1
  MAX_ITEM_SIZE                   1000
  OVERFLOW_CACHE_SIZE             0
  SUPPRESS_CACHE_ALERTS           YES

[LCG<TEST2MQ>]
  2PC_FOR_MESSAGES                NO
  2PC_FOR_REPORTINGS              NO
  APPLICATION_GROUP_NAME          SIAIP
  CACHE_CONTENT                   YES
  CACHE_NO                        777
  CACHE_SIZE                      150
  CARRIER_NAME                    SIAIP
  CHANNEL_TYPE                    IP-RT-MSG
  IMPORTER_COUNT                  2
  IP_JRN_WRITER                   ip_journal_writer; put lib here
  LCG_OWNER                       demofin:Bank1
  MAX_ITEM_SIZE                   1000
  MGTW_HOST                       ;
  OVERFLOW_CACHE_SIZE             0
  SUPPRESS_CACHE_ALERTS           YES
```

4.1.2 eximf002 Configuration

The library eximf002 is used for ISO20022 Instant Payment business message exchange.

SIAnet specific:

The SIAnet detailed configuration depends on the configured persistence level, esp. `DELIVERY_REPORT_GENERATION`. `SIAnet FEMS XS` requires two types of importers, if the persistence level is not '`InstPmtNoPersistence`':

- Unfiltered Importer for Receptions, possibly Technical Acks
- Filtered Importer for Notifications, which require server affinity

Parameter	DELIVERY_REPORT_ GENERATION	RESPONSE_QUEUE	SERVER_ RESPONSE_ MATCHING
SWIFT AGI	`DLV_REPORT_GEN_IMMEDIATE`	Not set	Not set (Off)
EBICS TRAVIC	`DLV_REPORT_GEN_IMMEDIATE`	Not set	Not set (Off)
FEMS XS (not persistent)	`DLV_REPORT_GEN_IMMEDIATE`	Not set	Not set (Off)
FEMS XS	`DLV_REPORT_GEN_REPLY`	Set to **Business User DATA Result Queue** (`INBOUND_QUEUE` to be set to Business User DATA Receive Queue)	`MsgId2CorrId`

Table 4.1-VII Parameter eximf002 Configuration

If using MQ-Cluster:

Please use the parameter `ADDITIONAL_CLUSTER_QMGR_LIST` to specify additional queue managers where queues using same name and function must exists.

4.2 Shared Data Source

The already mentioned Instant Payment Journal is based on a view of shared data source tables.

Please note, it has been shown, that establishing these tables in a separate database rather than integrating them into the BOX database is the preferred, hence recommended option.

For the installation of the external database please follow the database installation procedure described in document box_inst_v3r23.pdf.

4.2.1 Creating a JNDI Connection

The newly created database must be set as data source of the BOX database. A JNDI connection must be configured in the Workflow-Journals/Queues-Data Sources within a change set:

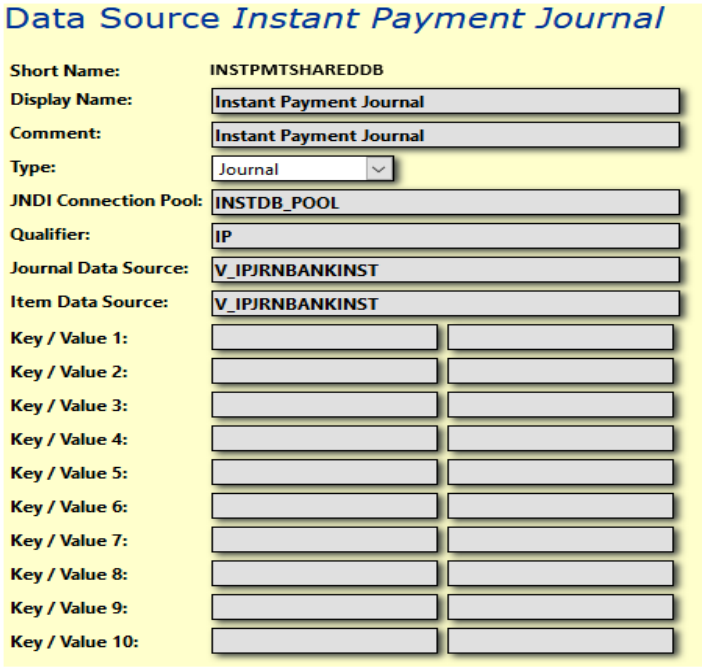


Figure 20 Creating a JNDI Connection

Figure 21 Configured Data Source

4.3 Adding Submission Flow Enrichment (Example SIA Flow List)

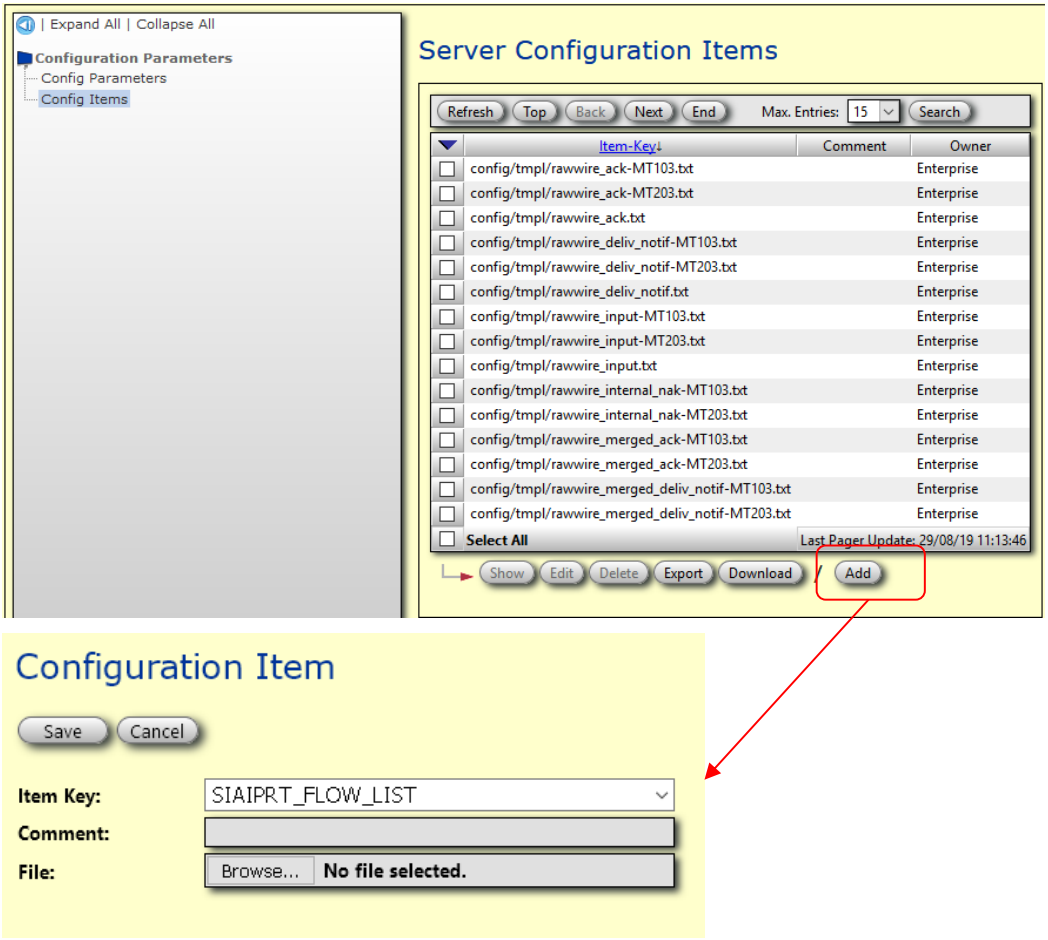


Figure 22 Adding Special Flow Table for SIA

5 Connectivity Channels

5.1 Connectivity Channel to SWIFT Network

The Connectivity Channel is a dedicated channel to connect to the SWIFT Network using the Alliance Gateway Instant (AGI) Plugin to receive and send SWIFT Instant Messages.

5.1.1 AGI Plugin (expgi_swift_agi)

The BOX Alliance Gateway Instant (AGI) Plugin is configured within the server configuration done using a Change Set of the BOX Web Client and is especially adapted to deal with Instant Payment messages.

5.1.1.1 Parameters

The expgi_swift_agi BOX Alliance Gateway Instant (AGI) Plugin has the following configuration parameters in the SWIFT_AGI_PLUGIN section:

Parameter	Default	Description
`LAU_KEY_1` `LAU_KEY_2`	At least one key must be configured	AES encrypted LAU key used for HMAC calculation/verification The AES encryption is done on the 32-character hexascii representation of the 16 bytes binary key. The keys have to match the LAU keys configured in SWIFT AGI.
`LAU_KEY_1_ID` `LAU_KEY_2_ID`	If `LAU_KEY_N` is configured in the plugin, `LAU_KEY_ID_N` must also be specified	ID of the keys configured in `LAU_KEY_1`, `LAU_KEY_2`, must match the values configured in SWIFT AGI.
`LAU_KEY_1_VALID_UNTIL` `LAU_KEY_2_VALID_UNTIL`	Date in `YYYY-MM-DD`	`LAU_KEY_1_VALID_UNTIL`, `LAU_KEY_2_VALID_UNTIL`: this parameters define the end of validity for `LAU_KEY_1`, `LAU_KEY_2`. If left empty, the validity of the key will never expire. The parameter can be set either to an ISO8601 timestamp or to a date in YYYY-MM-DD format. If set to a date, the date will be expanded to the end of the date in UTC time zone, e.g. 2019-01-31 will be expanded to 2019-01-31T23:59:59Z. If both `LAU_KEY_1` and `LAU_KEY_2` are set, `the LAU_KEY_1_VALID_UNTIL`, `LAU_LEY_2_VALID_UNTIL` parameters must be set to different values (keys cannot have same validity period). If both LAU keys are set the following rule applies for HMAC calculation/verification:

Parameter	Default	Description
		- If the actual time is before the valid until time of the oldest key, this key will be used for HMAC calculation/verification - if the actual time is after the valid until time of the oldest key but before the valid until parameter of the newest key, the newest key will be used.
LAU_KEY_OVERLAP_PERIOD	2	Overlap period in hours if both LAU_KEY_1, LAU_KEY_2 are configured (default 2 hours). This parameter allows a reverification of the HMAC of a received message with the other key if: - the verification with the oldest key failed but the actual time is mi the overlap period after the valid until time of the oldest key - the verification with the newest key failed but the actual time is in the overlap period before the valid until time of the oldest key
SUPPRESS_CACHE_ALERTS		This parameter [LCG<lcgname>].SUPPRESS_CACHE_ALERTS to was implemented to reduce system load caused by frequent alerts reporting critical cache state.

Table 5.1-VIII AGI Plugin Configuration Parameters

The following chapter shows a sample configuration of the Connectivity Channel and the AGI Plugin linked to this particular channel.

5.1.1.2 SWIFTnet Connectivity Channel Configuration Example

```
[LCG<IPSWIFT>]
  2PC_FOR_MESSAGES               NO
  2PC_FOR_REPORTINGS             NO
  APPLICATION_GROUP_NAME         IPSWIFT
  CACHE_CONTENT                  YES
  CACHE_SIZE                     200
  CGTW_HOST                      ; <Prococol>:ModuleID,LCG-Number
  CHANNEL_TYPE                   IP-RT
  DISABLE_LCG                    NO ; YES
  IMPORTER_COUNT                 32
  MAX_ITEM_SIZE                  1000
  OVERFLOW_CACHE_SIZE            0
  TRANSMITTER_BLOCKTHRESHOLD     0
  TRANSMITTER_COUNT              32
  TRANSMITTER_UNBLOCKTHRESHOLD   0
  TRANSPORT_COST                 0

[LCG<IPSWIFT>.PEXA]
  CREATOR_PREFIX                 TIPS
  DEFAULT_CREATOR                Intercope
  DEFAULT_EXCEPTION_SHORTLABEL   INSTP_XXX_SWI_014_Exception_Messages
  DEFAULT_IPS_SHORTLABEL         INSTP_XXX_SWO_013_Process_Incoming_Message
  DEFAULT_MPS_INITMODE           2; 1 - Instantiated, 2 - Pattern
  DEFAULT_REPORTING_SHORTLABEL   ; ReportingPattern2
  DELIVERY_MONITOR               YES
  DEVICE_TYPE                    0xF002
  IMPORT_CHECK_CYCLE             5
  MONITOR_CARRIER_DELIVERY       NO
```

```
    MPS_PERSISTENCE_LEVEL          FullPersistence
    STORAGE_PERIOD

  [LCG<IPSWIFT>.F002]
    DEFAULT_OUTBOUND_QUEUE           TO.SWIFT_AGI
    DEFAULT_OUTBOUND_QUEUE_MANAGER   $$R$SRV_QMGR_NAME$
    DEFAULT_REPLY_QUEUE              FROM.SWIFT_AGI; not used, but must be specified
    DEFAULT_REPLY_QUEUE_MANAGER      ; QM_ICHH2JK
    DELIVERY_REPORT_GENERATION       DLV_REPORT_GEN_IMMEDIATE
    EXCEPTION_BACKOUT_LEVEL          3
    GENERATE_COMMAND_REPORT          NO
    INBOUND_QUEUE                    FROM.SWIFT_AGI
    LOCAL_QUEUE_MANAGER              $$R$SRV_QMGR_NAME$
    MESSAGE_DUMP_LIMIT               100000
    MQ_USER_IDENTIFIER               mqm
    PLUGIN_LIBRARY_NAME              expgi_swift_agi
    TRASH_BACKOUT_LEVEL              2
    TRASH_QUEUE_NAME                 TRASH


[LCG<IPSWIFT>.F002.SWIFT_AGI_PLUGIN]
    LAU_KEY_1                        XXXXXXXXXX
    LAU_KEY_1_ID                     XXXXX
    SUPPRESS_LAU_VERIFICATION        YES
```

5.1.1.3 **Backend Connectivity**

The format of messages from a backend application can be either MQMD + file data or MQMD + RFH2 Header + file data.

If the backend application provides only MQMD and file data, the BOX Backend Application plug-in processes the data and generates a message in BOX XML format (MPS).
If the backend application provides an RFH2 Header, the BOX server creates an internal XML data structure that contains all name values of the message as children of the root node (canonical RFH2.xml).

This internal XML is of the form:
<RFH2>
<namevalue1> ... </namevalue1>
<namevalue2> ... </namevalue12>
<namevalue3> ... </namevalue3>
...
</RFH2>

The data from this internal XML is transformed into a message in BOX XML format (MPS) by means of XSLT. The message can then (optionally) be enriched with data from a Submission Profile (chapter2.5.1). The format of Output messages to be routed to a backend application is either MQMD + file data or MQMD + RFH2 Header + file data.

The format to be used depends on the backend application, i.e. on the format that the application expects.
If the backend application expects only MQMD and file data, the Backend Application LCG processes the BOX format message and hands it over to the backend application in the expected format.
If the backend application shall receive a message with RFH2 Header, the message in BOX XML format is transformed into an internal XML data structure by means of XSLT. And follows the format described above.

Please also refer to the ISO20022 Backend Application Plugin in box_plugins.pdf for further information on importing ISO20022 messages (MX, FACT, SWIFT/SIA T2S, SIA FTS) from a backend application as well as for exporting ISO20022 messages to a backend application.

5.1.1.4 **Lau Key Security**

- Checksum (HMAC) for RFH2 header data and / or payload data.
- No encryption of payload data
- Different LAU-Keys for different back-office applications for all messages and technical responses

Figure 23 LCG TIPS F002 configuration excerpt

5.2 Connectivity Channel to EBICS

5.2.1 Connectivity Channel Configuration Example

```
[LCG<TRAVIC_IP_IN_01>]
  CGTW_HOST                          ; <Prococol>:ModuleID,LCG-Number
  CHANNEL_TYPE                       IP-RT
  APPLICATION_GROUP_NAME             TRAVIC_IP_IN
  ; DEFAULT_DELIVERY_COMPOSITION     0x012101 // include origination report and owner
                                              attributes
  SUPPORTED_ADDRESSTYPES             IPRT_EB
  2PC_FOR_MESSAGES                   NO
  2PC_FOR_REPORTINGS                 NO
  CACHE_CONTENT                      YES
  CACHE_SIZE                         200
  MAX_ITEM_SIZE                      1000
  OVERFLOW_CACHE_SIZE                0
  IMPORTER_COUNT                     10
  TRANSMITTER_COUNT                  18

[LCG<TRAVIC_IP_IN_01>.PEXA]
   IMPORT_CHECK_CYCLE                2
   DEVICE_TYPE                       0xF002
   PEXA_LIBRARY                      eximf002_cl
   CREATOR_PREFIX                    $$R$PFX1$
   DEFAULT_CREATOR                   $$R$DEFAULT_CREATOR$
   DEFAULT_OWNER                     $$R$DEFAULT_CREATOR$
   DEFAULT_IPS_SHORTLABEL            INSTP_XXX_003_Process_Incoming_Message
   DEFAULT_MPS_INITMODE              2
   DEFAULT_EXCEPTION_LABELPREFIX     $$R$PFX1$
   DEFAULT_EXCEPTION_SHORTLABEL      INSTP_XXX_004_Exception_Messages
   DELIVERY_MONITOR                  NO
   MONITOR_CARRIER_DELIVERY          NO
   STORAGE_PERIOD                    24
   MPS_PERSISTENCE_LEVEL             NoPersistence

[LCG<TRAVIC_IP_IN_01>.F002]
   PLUGIN_LIBRARY_NAME               expgi_travic_ip
   LOCAL_QUEUE_MANAGER               $$R$QMGR$
   DEFAULT_OUTBOUND_QUEUE_MANAGER    $$R$QMGR$
   DEFAULT_OUTBOUND_QUEUE            $$R$TO.TRAVIC_IP$
   INBOUND_QUEUE                     $$R$FROM.TRAVIC_IP$
   DEFAULT_REPLY_QUEUE_MANAGER       $$R$QMGR$
   DEFAULT_REPLY_QUEUE               $$R$FROM.TRAVIC_IP$
   TRASH_QUEUE_NAME                  TRASH
   ;MQ_USER_IDENTIFIER               mqm
   MAX_MSGLEN_IN_GROUP               0
   DELIVERY_REPORT_GENERATION        0;4
                                      ; 0 // delivery report is submission report
                                      ; 1 // delivery report through COA
                                      ; 2 // delivery report through COD
                                      ; 3 // delivery report through PAN/NAN
                                      ; 4 // delivery report through reply
   MESSAGE_DUMP_LIMIT                100000
   GENERATE_COMMAND_REPORT           NO ; YES
   EXCEPTION_BACKOUT_LEVEL           10
   TRASH_BACKOUT_LEVEL               20
   MQ_MSG_PERSISTENT                 NO


[LCG<TRAVIC_IP_IN_01>.F002.TRAVIC_IP_PLUGIN]; no config parameters (yet)
```

5.3 VAN Gateway (EBICS/SWIFT) Configuration Options

5.3.1 EBICS

- Notification Request for receptions and a confirmation for outbound
- Notification Request on/off: TRAVIC configuration
- Confirmation: on/off by TRAVIC configuration
- Confirmation/ Error Reply: Delivery Notification handled as Technical ACK/NAK
  `DELIVERY_REPORT_GENERATION`: Immediate
- LCG in Central Server Module only,
- Use eximf002-InboundQueue to read Notification Request, Receptions, Confirmations, Error Replies
- Response importer not to be used
- `SERVER_RESPONSE_MATCHING OFF` (this is default)

5.3.2 SWIFT

- Notification Request on/off: AGI configuration
- Technical Ack: requested Always
- Notification: depending on parameter `NOTIFICATION_REQUIRED`: on Error for separate AGI-Notification Request-Queue, Always for unset AGI-Notification Request-Queue
- `DELIVERY_REPORT_GENERATION`: Immediate

AGI Adapter processing for AGI Technical Responses:

- Process positive Notify as Notification Request-Data
- Process negative Notify as Technical Ack
- Process SWIFT Technical Ack as BOX Technical Ack.
- LCG in Central Server module only,
- Use eximf002-InboundQueue to read Notification Request, Receptions, Notify,
- TechAckResponse importer not to be used
- `SERVER_RESPONSE_MATCHING` set to `OFF` (this is default)

5.4 Messaging Interface to SIA FemsXS

5.4.1 Exemplary Administration LCG per FEMS, containing 1 channel for each FEMS-BU

```
[LCG<FEMS01>]
  CHANNEL_TYPE    BOX-SIAIPRT-ADMIN
  MGTW_HOST       T:6
```

Figure 24 Exemplary SIA Channel Setup 1

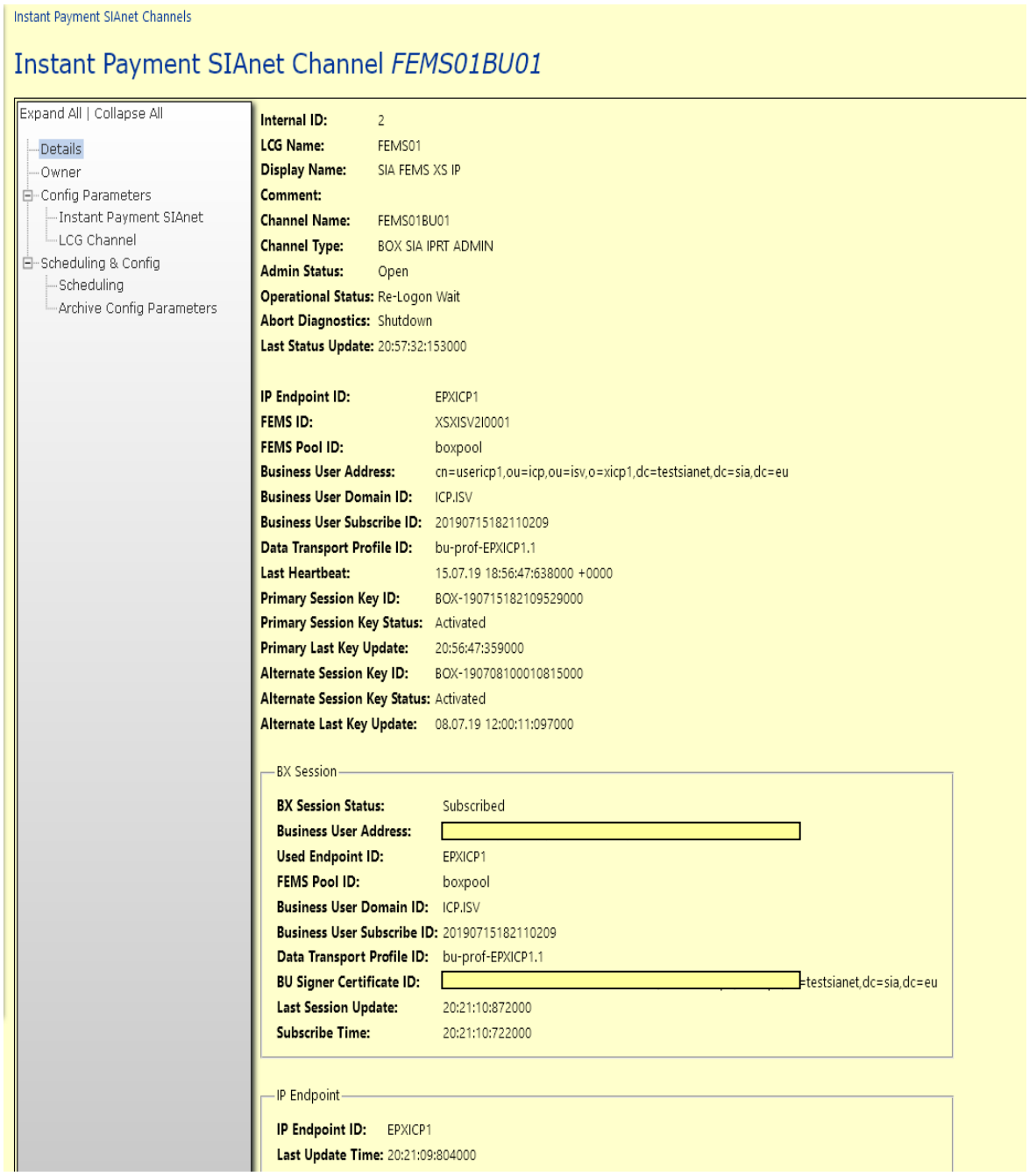


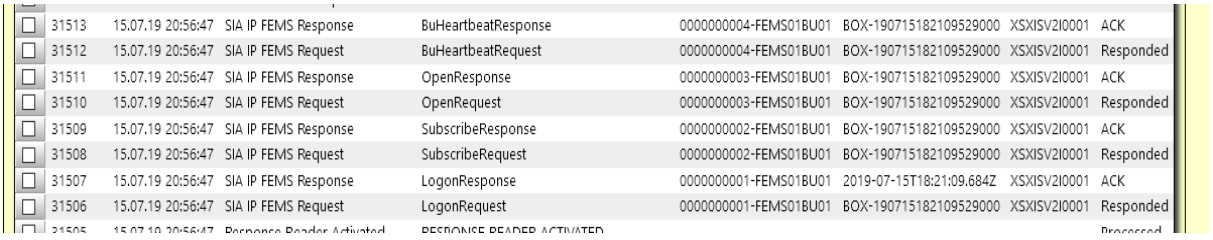
Figure 25 Exemplary SIA Channel Setup 2

5.4.1.1 Exemplary FEMS Administration LCG Configuration

```
[LCG<FEMS01>]
  CHANNEL_TYPE                   BOX-SIAIPRT-ADMIN
  CONTROL                        3; outbound only
  BLOCK_THRESHOLD                0
  UNBLOCK_THRESHOLD              0
  CHANNEL_LOAD                   20

[LCG<FEMS01>.CHAN<FEMS01BU01>]
  DEVICE_TYPE                   BOX-SIAIP ;
  CHANNEL_LIBRARY               mcimf215_mqcl
  CONTROL                       3

[FEMS01BU01]
  LOCAL_QUEUE_MGRNAME            XXXXXX
  TARGET_PLATFORM_ZOS            NO
  CMD_REQUEST_WRITE_QUEUE        TEST.CMD.REQ
  CMD_RESPONSE_READ_QUEUE        TEST.CMD.RSP
  INDICATION_READ_QUEUE          TEST.CMD.IND
  LAU_KEY_ID                     XXXXXX
  LAU_KEY                        XXXXXX
  END_POINT_ID                   XXXXXX
  FEMS_ID                        XXXXXX
  BUFEMSPOOL_ID                  boxpool
  BU_ADDRESS                     XXXXXX
  DOMAIN_ID                      XXXXXX
  DATA_TRANSPORT_PROFILE_ID      XXXXXX
  SIGNER_CERTIFICATE_ID          XXXXXX
  SIGNER_CERTIFICATE_PIN         XXXXXX
  INITIAL_ADMIN_CHANNEL_STATUS   Open ; Unchanged Closed LoggedIn Open
  COMMAND_TIMEOUT                30
  REOPEN_DELAY                   25
  MQ_EXPIRY_TIME                 45
  AUDIT_LEVEL                    0x00070000
  MESSAGE_DUMP_LIMIT             100000
```



31513	15.07.19 20:56:47	SIA IP FEMS Response	BuHeartbeatResponse	0000000004-FEMS01BU01	BOX-190715182109529000	XSXISV2I0001	ACK
31512	15.07.19 20:56:47	SIA IP FEMS Request	BuHeartbeatRequest	0000000004-FEMS01BU01	BOX-190715182109529000	XSXISV2I0001	Responded
31511	15.07.19 20:56:47	SIA IP FEMS Response	OpenResponse	0000000003-FEMS01BU01	BOX-190715182109529000	XSXISV2I0001	ACK
31510	15.07.19 20:56:47	SIA IP FEMS Request	OpenRequest	0000000003-FEMS01BU01	BOX-190715182109529000	XSXISV2I0001	Responded
31509	15.07.19 20:56:47	SIA IP FEMS Response	SubscribeResponse	0000000002-FEMS01BU01	BOX-190715182109529000	XSXISV2I0001	ACK
31508	15.07.19 20:56:47	SIA IP FEMS Request	SubscribeRequest	0000000002-FEMS01BU01	BOX-190715182109529000	XSXISV2I0001	Responded
31507	15.07.19 20:56:47	SIA IP FEMS Response	LogonResponse	0000000001-FEMS01BU01	2019-07-15T18:21:09.684Z	XSXISV2I0001	ACK
31506	15.07.19 20:56:47	SIA IP FEMS Request	LogonRequest	0000000001-FEMS01BU01	BOX-190715182109529000	XSXISV2I0001	Responded

Figure 26 Command Queue Set sharing (1 Set per FEMS)

5.4.1.2 SIA InstPmt Cache (SIA EP and BU/BX data sharing) in Central Server Module

```
[SIA_INSTPMT_CACHE]
  ENDPOINT_NAME                 XXXXXX
```

5.4.1.3 Exemplary FEMS Business Message LCG Configuration

- 1 Business Message LCG per BU,
- No channels on Messaging Interface Gateway;
- 1 Data queue set per BU

```
[LCG<BU01>]
  2PC_FOR_MESSAGES                  NO
  2PC_FOR_REPORTINGS                NO
  APPLICATION_GROUP_NAME            SIAIP
  CACHE_CONTENT                     YES
  CACHE_NO                          777
  CACHE_SIZE                        150
  CARRIER_NAME                      SIAIP
  CHANNEL_TYPE                      IP-RT-MSG
  IMPORTER_COUNT                    2
  IP_JRN_WRITER                     ip_journal_writer ; put lib here
  LCG_OWNER                         demofin:Bank1
  MAX_ITEM_SIZE                     1000
  MGTW_HOST                          ;
  OVERFLOW_CACHE_SIZE               0
  SUPPRESS_CACHE_ALERTS             YES
  TRANSMITTER_BLOCKTHRESHOLD        0
  TRANSMITTER_COUNT                 5
  TRANSMITTER_UNBLOCKTHRESHOLD      0
  TRANSPORT_COST                    0

[LCG<BU01>.PEXA]
  CREATOR_PREFIX                    demofin
  DEFAULT_CREATOR                   FINdivision
  DEFAULT_EXCEPTION_SHORTLABEL      IP_EXCEPTION
  DEFAULT_IPS_SHORTLABEL            IPSIA_INFLOW
  DEFAULT_MPS_INITMODE              2 ; 1 - Instantiated, 2 - Pattern
  DEFAULT_REPORTING_SHORTLABEL      ;ReportingPattern2
  DELIVERY_MONITOR                  YES
  DEVICE_TYPE                       0xF002
  IMPORT_CHECK_CYCLE                5
  MONITOR_CARRIER_DELIVERY          NO
  MPS_PERSISTENCE_LEVEL             InstPmtJrnPersistence ;FullPersistence ;
InstPmtJrnPersistence; InstPmtNoPersistence ; InstPmtMPSPersistence;
  PEXA_LIBRARY                      eximf002_cl
  STORAGE_PERIOD                    12

[LCG<BU01>.F002]
  RESPONSE_QUEUE                    TEST.RESPONSE.FEMS.QUEUE
  DEFAULT_OUTBOUND_QUEUE            TEST.OUTBOUND.FEMS.QUEUE
  DEFAULT_OUTBOUND_QUEUE_MANAGER    XXXXXXX
  DEFAULT_REPLY_QUEUE               TEST.REPLY.FEMS.QUEUE
  DEFAULT_REPLY_QUEUE_MANAGER       XXXXXXX
  DELIVERY_REPORT_GENERATION        DLV_REPORT_GEN_REPLY
  EXCEPTION_BACKOUT_LEVEL           10
  GENERATE_COMMAND_REPORT           NO
  INBOUND_QUEUE                     TEST.INBOUND.FEMS.QUEUE
  LOCAL_QUEUE_MANAGER               XXXXXXX
  MESSAGE_DUMP_LIMIT                10000
  MQ_USER_IDENTIFIER                 ;
  PLUGIN_LIBRARY_NAME               expgi_sianet_fems
  SERVER_RESPONSE_MATCHING          MsgId2CorrId
  TRASH_BACKOUT_LEVEL               20
  TRASH_QUEUE_NAME                  TEST.TRASH.FEMS.QUEUE
```

```
[LCG<BU01>.F002.SIANET_FEMS_PLUGIN]
  BUSINESS_USER_ADDRESS            XXXXXXXX
```

5.4.1.4 Configuration Options

- No extra Notification Request (included with Notify/Reception), Notify, Technical Ack: Notify – `DeliveryToCarrier`
- Technical Ack: Delivery Notification
- Data LCG in Central Server Module,
- Notify and Technical Ack read by filtered response queue importer
- Data LCG per Business User (BU), 1 Data Queue set per BU (shared by all FEMS/Central server)
- Persistence not InstPmtNoPersistence
- `SERVER_RESPONSE_MATCHING` set to `MsgId2CorrId`

else

- `SERVER_RESPONSE_MATCHING` set to `OFF` (this is default)

SIA Artefacts / Concepts:

- Endpoint: 1 Endpoint per BOX (defined through Instant Payment Database)
- Business User Address (BU, BX session): n per BOX
- FEMS XS instances: serving an Endpoint and multiple BUs
- XS Pool: Load balancing for a BU

6 OFAC Check Integration

OFAC is the Office of Foreign Asset Control, part of the U.S. Department of Treasury. OFAC is responsible for administering and enforcing economic and trade sanctions against certain nations, entities and individuals. OFAC maintains a listing of these restricted counter parties in a document called the "Specially Designated Nationals List" (SDN).

The BOX OFAC Check Integration uses IBM WebSphere MQ and is included in the BOX workflow.

6.1 Asynchronous Communication

Architectural Overview

SWIFT input messages are routed by the BOX workflow to an “OFAC Check Waiting Queue“.

A CPI with Custom Mode “OFACCheck” writes these messages to an MQ queue. The OFAC application reads these messages and writes either a SWIFT NAK or the original message to the ReplyToQueue. The result of the check is stored with the message. Based on this information BOX decides whether the message is further processed or interrupted.

6.2 Workflow Concept

New IPS “Forward to OFAC”

Based on the workflow BOX decides whether messages must be routed to the OFAC check. For this purpose, a new IPS “Forward to OFAC“ is implemented consisting of:

- An SBI (Analysis 1) takes the decision if the OFAC check has to be executed
- A CPI Writes the message to an MQ queue and waits for the result which is appended to the message as report.
- An SBI (Analysis 1) Checks the report. If the message is rejected it is either routed with a TGI to an application queue “Declined by OFAC“ or sent back with a DLI to the backend application as “merged Ack” (generated by the backend channel).

6.2.1 Exemplary Workflow of the OFAC Integration

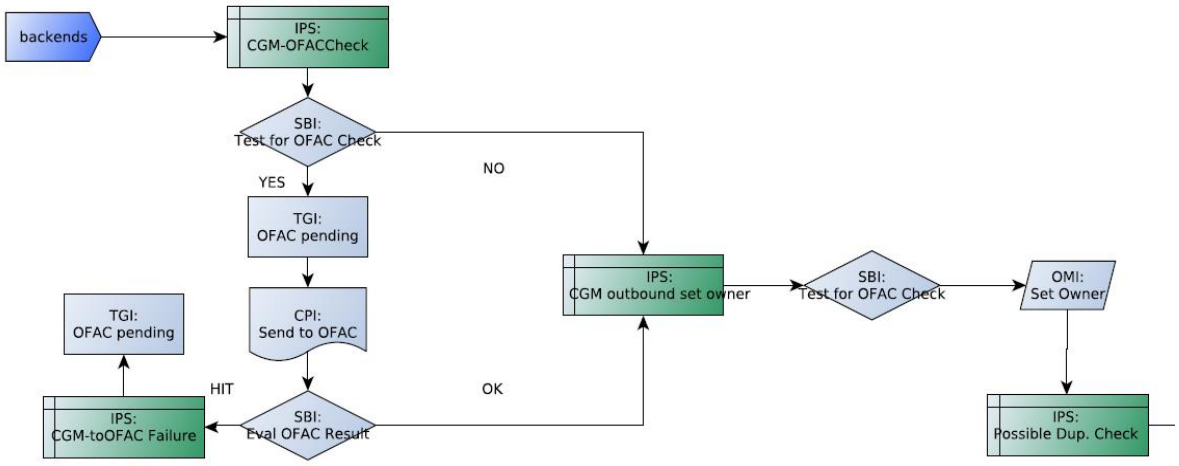


Figure 27 Exemplary OFAC Integration workflow

6.2.2 The Send to SWIFT or Reject

Application Queue “Declined by OFAC“

Messages which have been routed to the “Declined by OFAC“ queue are manually processed including the following operations:

- Forward to SWIFT, Release the message and continue processing
- Route to Backend, Reject the message and send “Merged Ack“ to the backend application.
- View OFAC Result Visualization of the message together with the result of the OFAC check

6.3 The Interrupt OFAC Check

Application Queue “Wait For OFAC“

Three operations (tasks) are provided for messages queued in “Wait For OFAC“:

- Interrupt OFAC check (multi selection possible)

  The asynchronous CPI is immediately terminated and a report (interrupted by operator) appended to the message. This report has the same format as a report generated by the OFAC check. A subsequent SBI (Analysis 1) decides if the message is sent or routed to a backend application

- MPS Details

  Shows details of the message

- Show

  Shows the payload of the message

- The IPS “RouteToWBIFN_ACK“ is extended by an SBI (Analysis 1) analysing the result of the OFAC check. If the message has not yet been verified it is routed to the application queue „OFAC Check after transmission“.

6.4 Check of Already Sent Messages

The application queue “OFAC check after transmission”

Messages which have been sent without a valid OFAC check are routed to the queue “OFAC check after transmission”. A CPI writes the message to a MQ queue and waits for the result of the OFAC check. The following two operations (tasks) are provided for this queue:

- MPS Details Shows details of the message

- Show Shows the payload of the message

6.5 Message Enrichment

Message format extensions

The result of the OFAC check is stored in the following folder:

```
<meadow>
<OFACVaildationData>
</OFACVaildationData>
</meadow>
```

The exact structure of the folder will be defined during development:

```
<meadow>
<OFACValidationData>
<MessageValidationStatus numVal=”2”>Valid</MessageValidationStatus>
<MessageValidationDescription>Message is
valid</MessageValidationDescription>
</OFACValidationData>
</meadow>
```

6.6 Interfaces

The interface between BOX and the OFAC application is IBM WebSphere MQ. All messages, which are to be checked are written to an MQ queue. A temporary queue with a dynamic queue name is specified as ReplyToQueue. The queue name is unique for each message forwarded to the OFAC application. When the OFAC application rejects a message, it sends back a pseudo SWIFT NAK. If the messages pass the OFAC check the message is sent back in wire format.

7 Manual Message Entry for Tests

7.1 Pacs.008.001.02

The message Pacs.008.001.02 is used to transport the Payment instruction from the Originator Bank to the Beneficiary Bank, directly or through intermediaries. The message caters for bulk and single payment instructions.

For manual testing, a Pacs.008.001.02 message is written to a backend queue, which is read by BOX.

The general structure of a test message is

```
<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns="urn:iso:std:iso:20022:tech:xsd:pacs.008.001.02"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:iso:std:iso:20022:tech:xsd:pacs.008.001.02
../xmlschemas/EPC122-16_2017_V1.1_pacs.008.001.02.xsd">
<FIToFICstmrCdtTrf>
<GrpHdr>
   <MsgId>${individually generated by Originator}</MsgId>
   <CreDtTm>${individually generated, Time of message generation}</CreDtTm>
   <NbOfTxs>1</NbOfTxs>
   <TtlIntrBkSttlmAmt Ccy="EUR">${individually generated amount}</TtlIntrBkSttlmAmt>
   <IntrBkSttlmDt>${individually generated}</IntrBkSttlmDt>
   <SttlmInf>
          <SttlmMtd>CLRG</SttlmMtd>
          <ClrSys><Prtry>IPS</Prtry></ClrSys>
   </SttlmInf>
   <PmtTpInf>
          <SvcLvl><Cd>SEPA</Cd></SvcLvl>
          <LclInstrm><Cd>INST</Cd></LclInstrm>
   </PmtTpInf>
   <InstgAgt><FinInstnId><BIC></BIC></FinInstnId></InstgAgt>
   <InstdAgt><FinInstnId><BIC></BIC></FinInstnId></InstdAgt>
</GrpHdr>
<CdtTrfTxInf>
   <PmtId>
          <InstrId>${individually generated and optional}</InstrId>
          <EndToEndId>${individually generated by Originator, identifies the SCT
Transaction}</EndToEndId>
          <TxId>${individually generated by Originator, identifies the SCT
Transaction}</TxId>
   </PmtId>
   <IntrBkSttlmAmt Ccy="EUR">${individually generated, amount}</IntrBkSttlmAmt>
   <AccptncDtTm>${individually generated, reception time of the SCT Transaction,
Originator}</AccptncDtTm>
   <ChrgBr>SLEV</ChrgBr>
   <Dbtr><Nm>${individually generated, Originator}</Nm></Dbtr>
   <DbtrAcct><Id><IBAN>${individually generated, account of
Originator}</IBAN></Id></DbtrAcct>
   <DbtrAgt><FinInstnId><BIC></BIC></FinInstnId></DbtrAgt>
   <CdtrAgt><FinInstnId><BIC></BIC></FinInstnId></CdtrAgt>
   <Cdtr><Nm>${individually generated, Beneficiary}</Nm></Cdtr>
   <CdtrAcct><Id><IBAN>${individually generated, Beneficiary
account}</IBAN></Id></CdtrAcct>
</CdtTrfTxInf>
</FIToFICstmrCdtTrf>
</Document>
```

8 Workflow

8.1 Exemplary SWIFTnet Workflow of an Archive-Persistence Mode

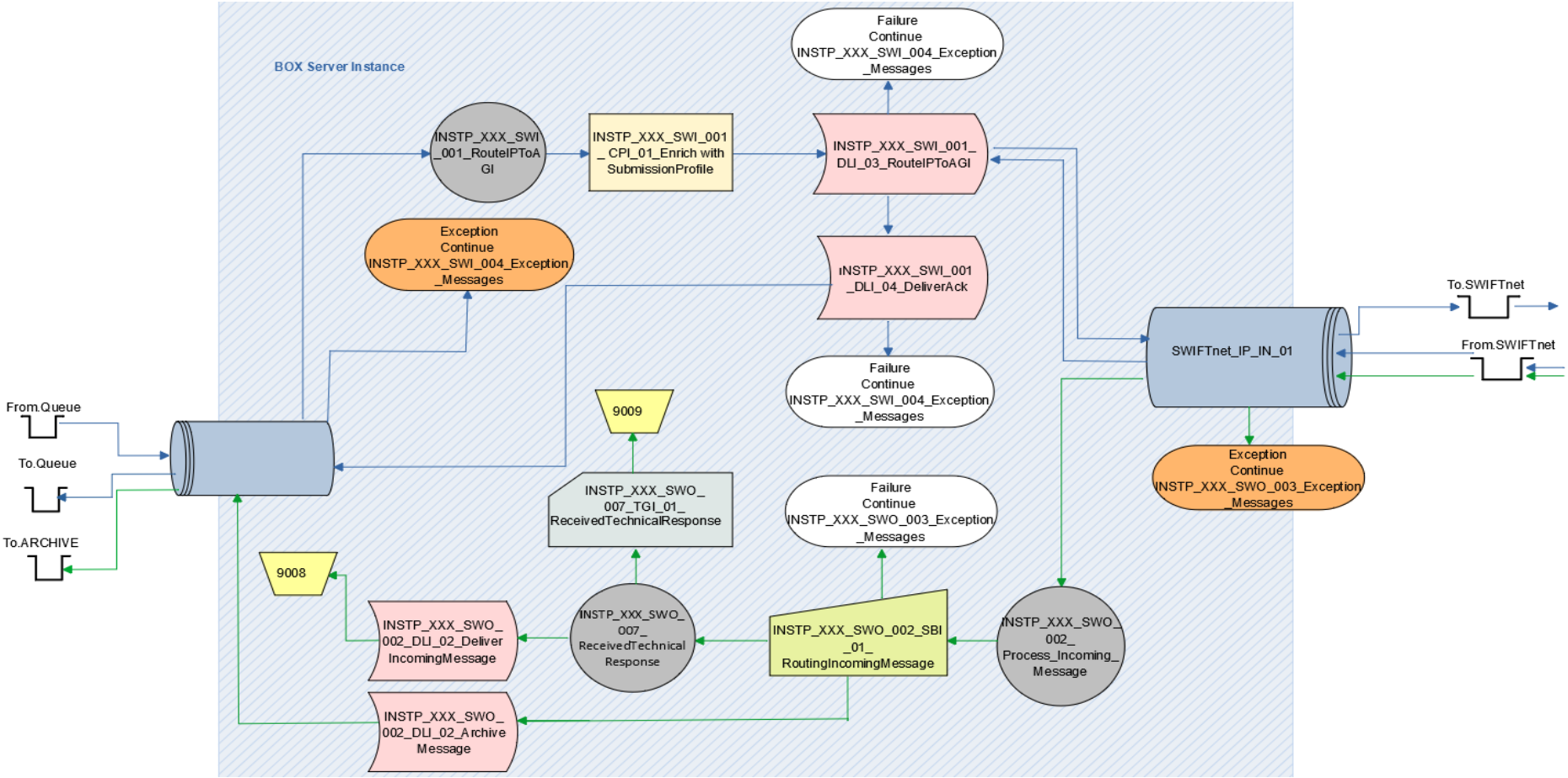


Figure 28 Overview SWIFTnet Instant Payment Workflow (TIPS)

8.2 Message Exception Workflow

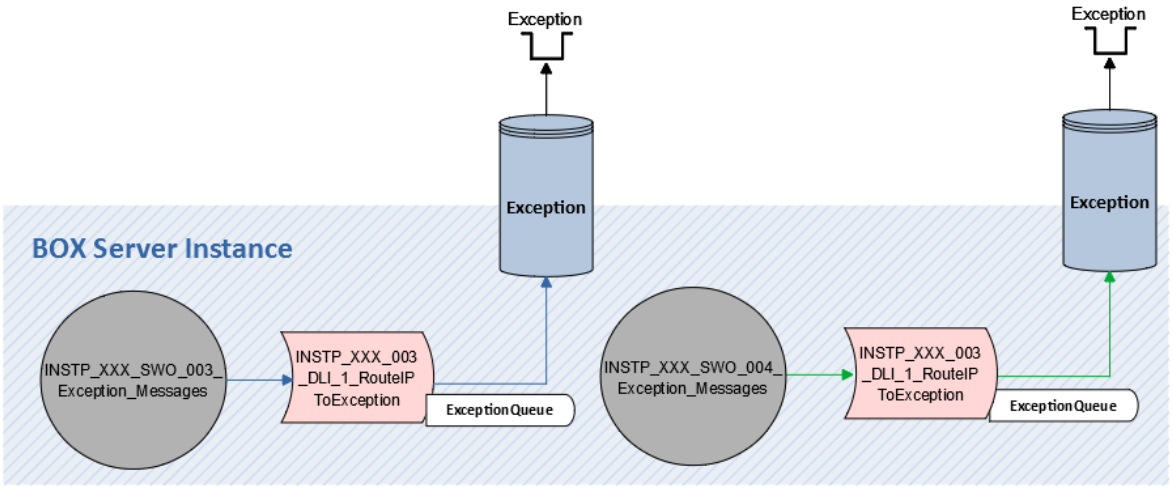


Figure 29 Overview Exception Message Workflow

8.3 Signs and Symbols

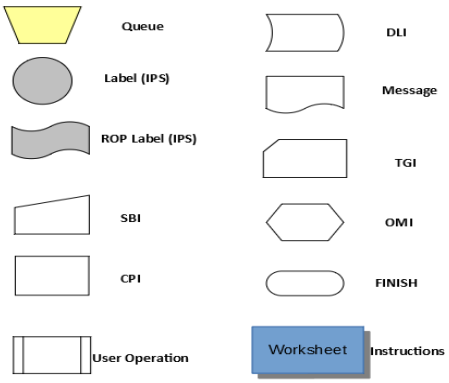


Figure 30 Workflow Signs and Symbols

8.4 Instruction Patterns

8.4.1 Outgoing

```
INSTP_XXX_SWO_002_Process_Incoming_Message
INSTP_XXX_SWO_002_SBI_01_ RoutingIncomingMessage
INSTP_XXX_SWO_007_ReceivedTechnicalResponse
INSTP_XXX_SWO_007_TGI_01_ReceivedTechnicalResponse
INSTP_XXX_SWO_002_DLI_02_Deliver_IncomingMessage
INSTP_XXX_SWO_003_Exception_Messages
```

8.4.2 Incoming

```
INSTP_XXX_SWI_001_RouteIPToAGI
INSTP_XXX_SWI_001_CPI_01_Enrich (see graph below)
INSTP_XXX_SWI_001_DLI_03_RouteIPToAG
```

INSTP_XXX_SWI_001_DLI_04_DeliverAck
INSTP_XXX_SWI_004_Exception_Messages

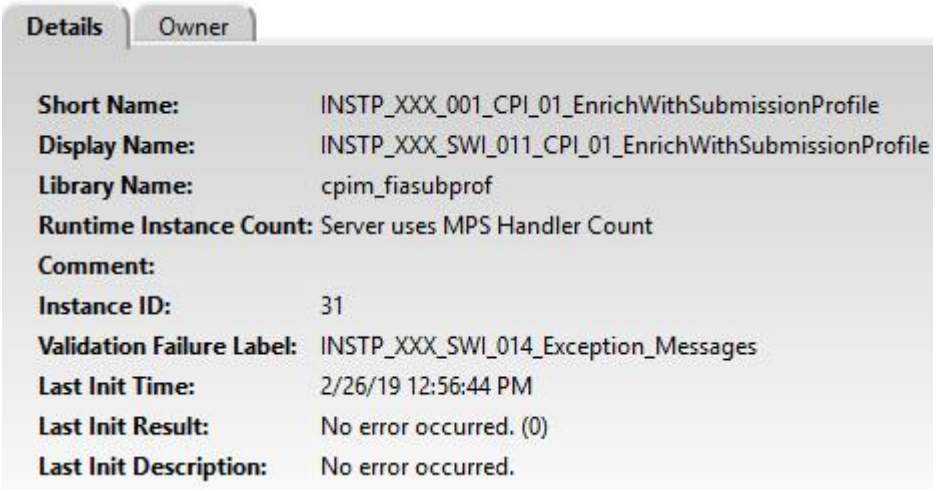


Figure 31 Content Processing Modules-FIA Subm. Prof. Instance: Enrichment with Submission Profile

8.4.3 Analysis 1

Analysis 1 provides a C-like programming language, which can be used to analyse various data objects, which are part of a Message Processing Sequence in BOX, such as MPS General Attributes, Report data and Content Version.
Analysis 1 is used during the 'Outgoing' Instant Payments workflow to determine, whether the message reflects a technical response, or an Instant Payment message received from SWIFTnet to be routed to the Payment Application.

Within Analysis 1 decisions are made following the either/or and if/then pattern.

The following is taken from the Workflow:

```
INSTP_XXX_SWO_002_Process_Incoming_Message

CheckForTechnicalResponse
if ( 4 == mpsga( ADTYPE)){
    print(" Technical Response received ");
    setips INSTP_XXX_SWO_007_Received_Technical_Response;
  }
RoutingIncomingMessage
mq_address = getreplacementtokenvalue
("IAFABA:$$R$TO_PAYAPP_01$@$$R$SRV_QMGR_NAME$");
addtosimpleaddrlist(mq_address);
return;
```

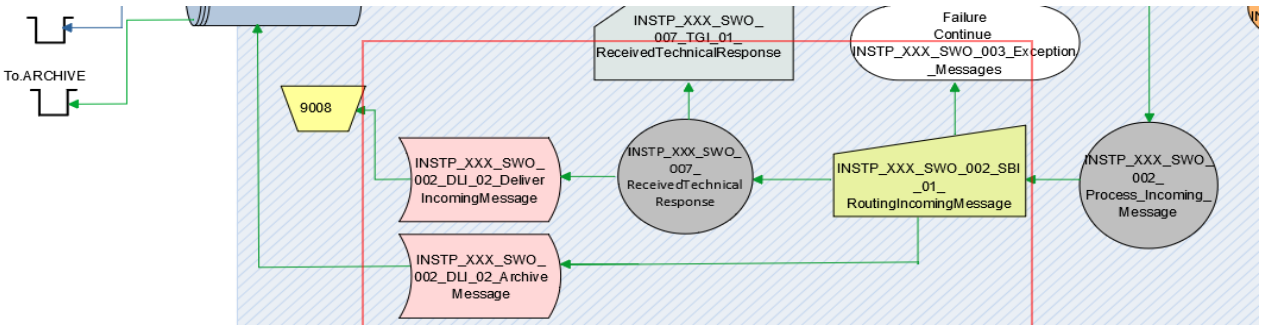


Figure 32 Excerpt Workflow containing Analysis 1

9 Monitoring BOX with SNMP Dashboard

9.1 Monitoring BOX in non-persistence mode

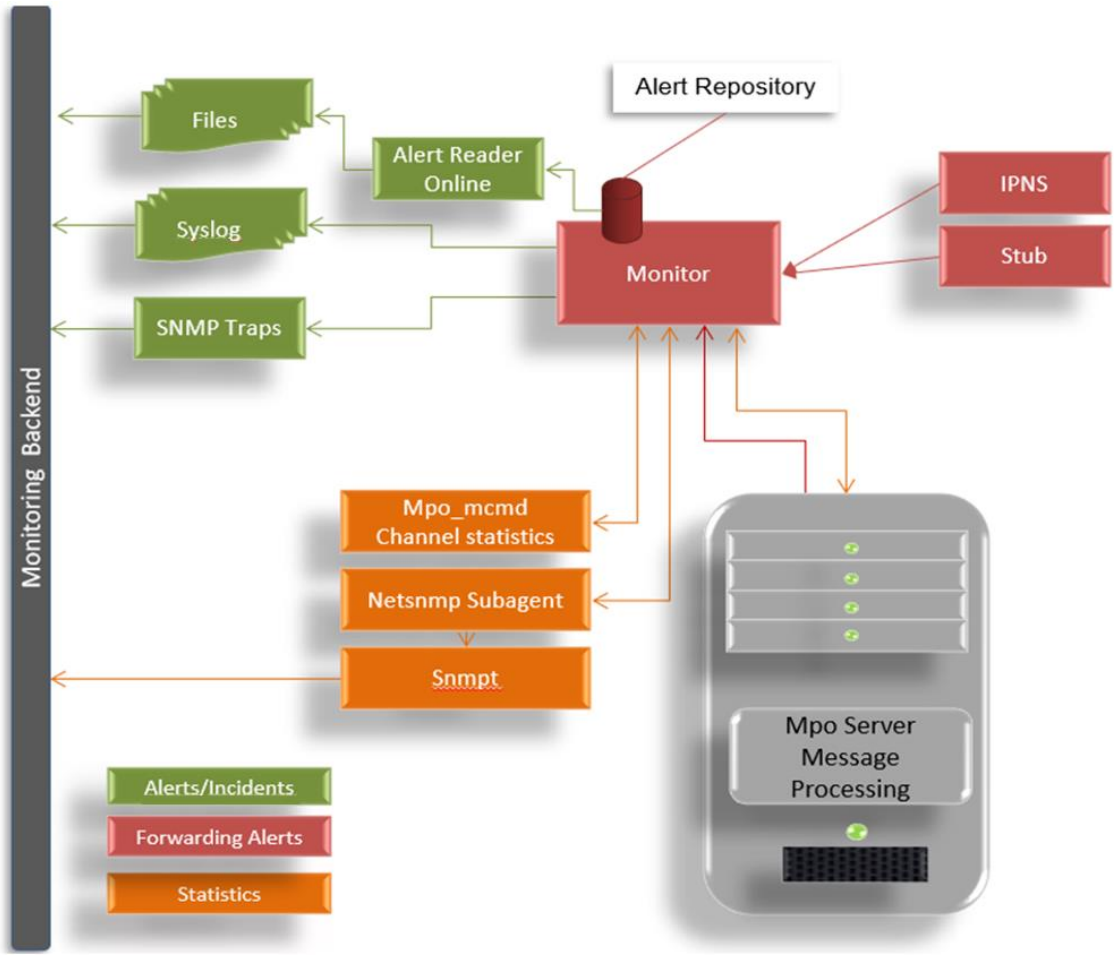


Figure 33 Overview Monitoring in Non-Persistence Mode

9.1.1 System Monitor Module

The System Monitor Module controls and monitors the domain. It starts all required components it finds in the configuration of the domain and continuously receives heartbeat signals from all active components. If a component fails, it is automatically restarted by the System Monitor. When a domain is shutdown the System Monitor sends a signal to all components so they can stop operations in a controlled way. It uses the stub module to start and stop components and receive alerts (error-, warning-, and information messages) from all modules.

These alerts can either be forwarded and translated into SNMP traps, which in turn are read by an SNMP monitoring backend (respective MIB provided by Intercope) or analysed by alert files read by the Alert Reader tool also provided by Intercope. It is also possible to analyze the respective syslog.

9.1.2 Monitor Command Tool mpo_mcmd

With the Monitor Command Tool (mpo_mcmd) you can set the administrative status of the BOX modules and perform status queries against the modules. The administrative status (AdminStatus) refers to the desired status of a module, while the operational status (OperStatus) refers to the actual status of the module. For further details, please refer to the document box_admig_vXrXX.pdf.

Example

```
mpo_mcmd 1 /I1 /MB3 /G
```

Server LCG Name Host	AdminStatus	OperStatus	LastChange
PTSADES0X 000273	active	unknown	Fri May 29 12:22:19 2015
Export	active	active	
Import	active	active	
PTSADESS_IAFA 000373	disabled	unknown	Fri May 29 12:22:13 2015
Export	active	active	
Import	active	active	
FACTBA_1 0006B3	active	unknown	Fri May 29 12:22:20 2015
Export	active	active	
Import	active	active	
SEPASTATUS 0006B3	active	unknown	Fri May 29 12:22:20 2015
Export	active	active	
Import	active	active	
SIAT2S 000473	disabled	unknown	Fri May 29 12:22:13 2015
Export	active	active	
Import	active	active	

9.1.3 SNMP Integration with Zabbix

To enable a graphical display of the server's health, Zabbix is a tool, which can be used. Please contact Intercope for support.

Zabbix, an open source monitoring solution created by Alexei Vladishev, is currently actively developed and supported by Zabbix SIA. It is written and distributed under the GPL General Public License version 2.

It monitors parameters of a network and the health and integrity of servers. It uses a flexible notification mechanism and allows the configuration of e-mail-based alerts for events. It provides reporting and data visualization features based on stored data and supports both polling and trapping.

Through the Zabbix web-based frontend, reports, statistics and configuration parameters are accessible.

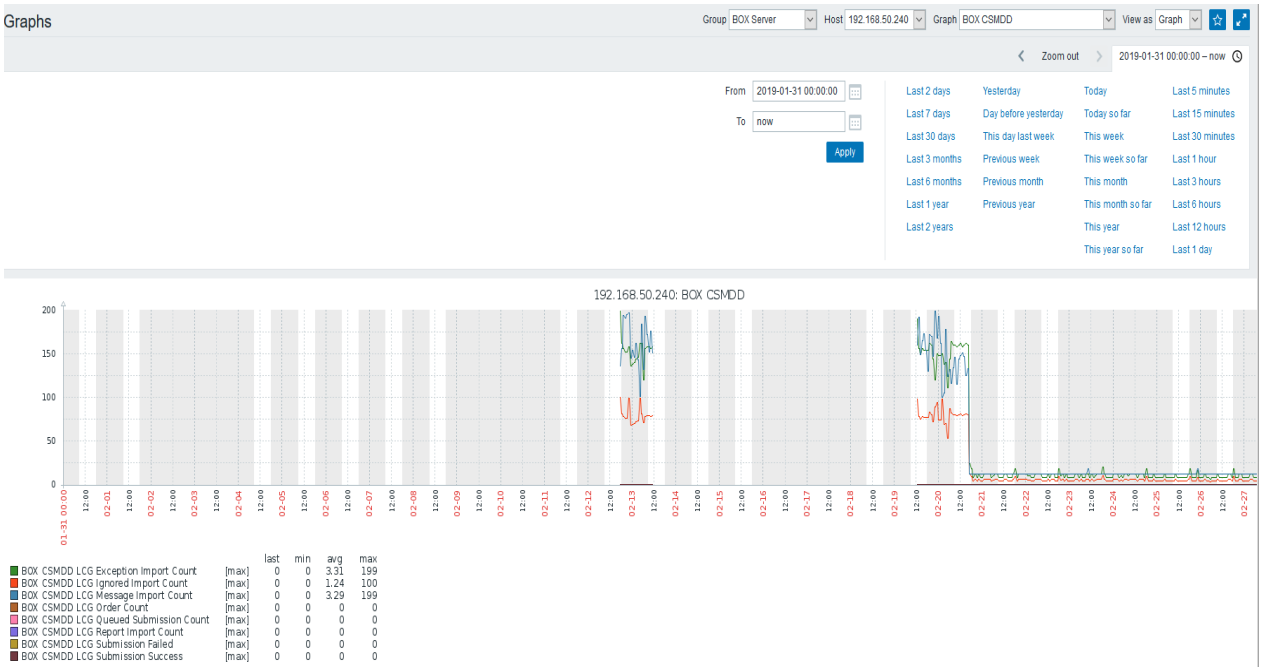


Figure 34 BOX Server Health Monitoring Example

10 Appendix

10.1 Parameter

10.1.1 Analysis1

The following aliases have been defined in Analysis1 to retrieve SWIFT instant payment related data:

Please note, that technical responses on input messages (SWIFTNet: Notify and TechnicalAck) use ApplicationDefinedType 4 (Technical Response) in GenericAttributeSet.

CV origination report:	
CV_OR_PROTREP_IPRTSW_MSG_REF	Message reference, alias for TEXT 511,1
CV_OR_PROTREP_IPRTSW_ADDITIONAL_INFO	Additional info, alias for TEXT 511, 2
CV_OR_PROTREP_IPRTSW_REQUESTOR_DN	Requestor DN, alias for TEXT 255, 1
CV_OR_PROTREP_IPRTSW_RESPONDER_DN	Responder DN, alias for TEXT 255, 2
CV_OR_PROTREP_IPRTSW_SERVICE_NAME	Service name, alias for TEXT63, 1
CV_OR_PROTREP_IPRTSW_MSG_TYPE	Message type, alias for TEXT 63, 2
CV_OR_PROTREP_IPRTSW_MSG_NETWORK_REF	Message network reference, alias for TEXT 63, 5
CV_OR_PROTREP_IPRTSW_CHANNEL_NAME	Channel name, alias for TEXT 63, 7
CV_OR_PROTREP_IPRTSW_PROTOCOL_CODE	Protocol code, alias for TEXT 63, 9
CV_OR_PROTREP_IPRTSW_DEVICE_CODE	Device code, alias for TEXT 63, 10
CV_OR_PROTREP_IPRTSW_POSSIBLE_DUPLICATE	Possible duplicate, alias for NUM 4
CV_OR_PROTREP_IPRTSW_SEND_TIME	Send time, alias for TIME 1
CV_OR_PROTREP_IPRTSW_RECEIVE_TIME	Receive time, alias for TIME 2
SDA report:	
SDA_PROTREP_IPRTSW_MSG_REF	Message reference, alias for TEXT 511,1
SDA_PROTREP_IPRTSW_ADDITIONAL_INFO	Additional info, alias for TEXT 511, 2
SDA_PROTREP_IPRTSW_PRIMITIVE_ERROR_TEXT	Primitive error text, alias for TEXT 511, 3
SDA_PROTREP_IPRTSW_REQUESTOR_DN	Requestor DN, alias for TEXT 255, 1
SDA_PROTREP_IPRTSW_RESPONDER_DN	Responder DN, alias for TEXT 255, 2
SDA_PROTREP_IPRTSW_SERVICE_NAME	Service name, alias for TEXT63, 1
SDA_PROTREP_IPRTSW_MSG_TYPE	Message type, alias for TEXT 63, 2
SDA_PROTREP_IPRTSW_MSG_NETWORK_REF	Message network reference, alias for TEXT 63, 5
SDA_PROTREP_IPRTSW_CHANNEL_NAME	Channel name, alias for TEXT 63, 7
SDA_PROTREP_IPRTSW_REPORT_SOURCE	Report source, alias for TEXT 63, 8
SDA_PROTREP_IPRTSW_PROTOCOL_CODE	Protocol code, alias for TEXT 63, 9

SDA_PROTREP_IPRTSW_DEVICE_CODE	Device code, alias for TEXT 63, 10
SDA_PROTREP_IPRTSW_PRIMITIVE_RETURN_CODE	Primitive return code, alias for TEXT 15, 1
SDA_PROTREP_IPRTSW_POSSIBLE_DUPLICATE	Possible duplicate, alias for NUM 4
SDA_PROTREP_IPRTSW_SEND_TIME	Send time, alias for TIME 1
SDA_PROTREP_IPRTSW_RECEIVE_TIME	Receive time, alias for TIME 2
ISDREP report;	
ISDREP_PROTREP_IPRTSW_MSG_REF	Message reference, alias for TEXT 511,1
ISDREP_PROTREP_IPRTSW_ADDITIONAL_INFO	Additional info, alias for TEXT 511, 2
ISDREP_PROTREP_IPRTSW_PRIMITIVE_ERROR_TEXT	Primitive error text, alias for TEXT 511, 3
ISDREP_PROTREP_IPRTSW_REQUESTOR_DN	Requestor DN, alias for TEXT 255, 1
ISDREP_PROTREP_IPRTSW_RESPONDER_DN	Responder DN, alias for TEXT 255, 2
ISDREP_PROTREP_IPRTSW_SERVICE_NAME	Service name, alias for TEXT63, 1
ISDREP_PROTREP_IPRTSW_MSG_TYPE	Message type, alias for TEXT 63, 2
ISDREP_PROTREP_IPRTSW_MSG_NETWORK_REF	Message network reference, alias for TEXT 63, 5
ISDREP_PROTREP_IPRTSW_CHANNEL_NAME	Channel name, alias for TEXT 63, 7
ISDREP_PROTREP_IPRTSW_REPORT_SOURCE	Report source, alias for TEXT 63, 8
ISDREP_PROTREP_IPRTSW_PROTOCOL_CODE	Protocol code, alias for TEXT 63, 9
ISDREP_PROTREP_IPRTSW_DEVICE_CODE	Device code, alias for TEXT 63, 10
ISDREP_PROTREP_IPRTSW_PRIMITIVE_RETURN_CODE	Primitive return code, alias for TEXT 15, 1
ISDREP_PROTREP_IPRTSW_POSSIBLE_DUPLICATE	Possible duplicate, alias for NUM 4
ISDREP_PROTREP_IPRTSW_SEND_TIME	Send time, alias for TIME 1
ISDREP_PROTREP_IPRTSW_RECEIVE_TIME	Receive time, alias for TIME 2
UPM and address book:	
UPMADR_IPRTSW_RESPONDER_DN	Responder DN, alias for TEXT 255, 2
ABRECADR_IPRTSW_RESPONDER_DN	Responder DN, alias for TEXT 255, 2

Table 10.1-IX Analysis 1 Parameter

MQ BACKOUT COUNT

Address Type	Parameter
MQ	CV_OR_PROTREP_MQ_BACKOUT_COUNT
IPRT_EB	CV_OR_PROTREP_IPRTEB_MQ_BACKOUT_COUNT

Table 10.1-X MQ Backout Parameter

11 Disclaimer

INTERCOPE International Communication Products Engineering GmbH (Intercope) and the stylized logo is the registered trademark of Intercope and its subsidiaries, in Germany and certain other countries. All other trademarks mentioned in this document are the acknowledged property of their respective owners.

Intercope provides this publication "as is" without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of non-infringement, merchantability or fitness for a particular purpose.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. Intercope may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

This information may contain sample application programs in source language, which illustrate programming and implementation techniques. You may copy, modify, and distribute these samples programs in any form without payment to Intercope, for the purposes of developing, using, marketing or distributing application programs for which the sample programs are written. These examples have not been thoroughly tested under all conditions. Intercope, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.
The sample programs are provided "AS IS", without warranty of any kind. Intercope shall not be liable for any damages arising out of use of the sample programs.

Intercope grants the right to reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. Intercope does not allow derivative works of these publications, or to reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of Intercope.

Without written permission of Intercope no part of this publication may be modified and/or reproduced in any way.

INTERCOPE GmbH

Himmelstrasse 12-16,

22299 Hamburg,

Germany

+49 40 514 52 0

info@intercope.com

https://www.intercope.com