INTERCOPE

Version 3 Release 27

BOX Messaging Hub Instant Payment

Concept and Implementation

Table of Contents

List of Tables

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.

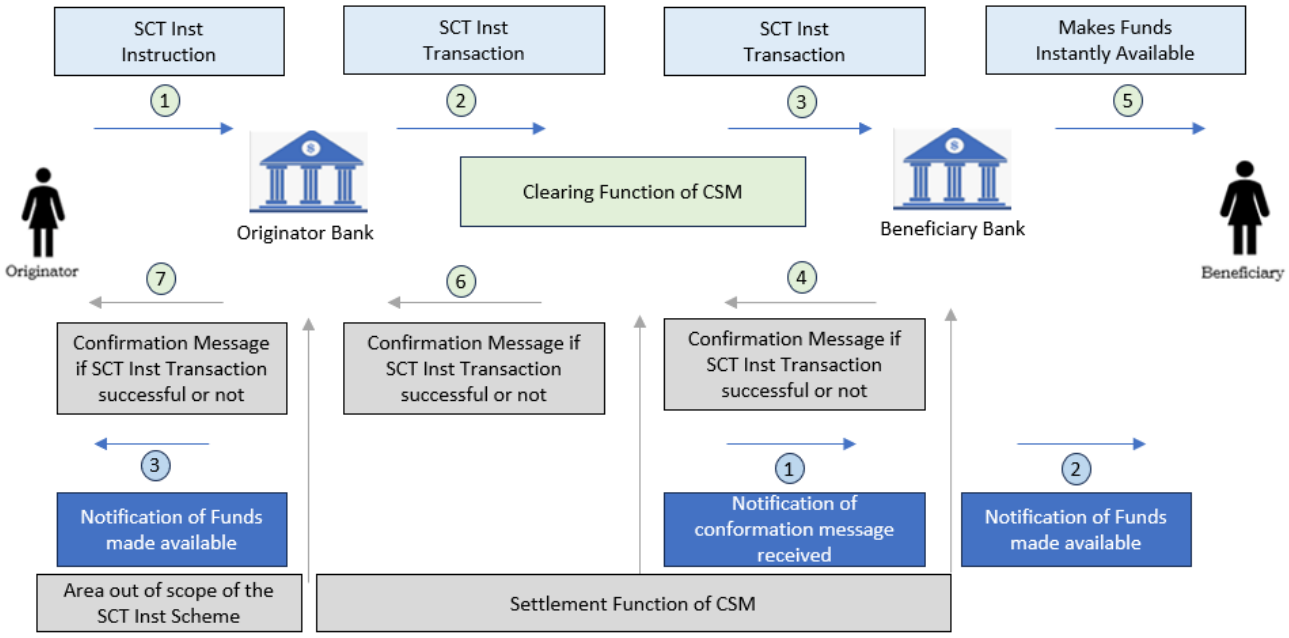
Irrevocability

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.



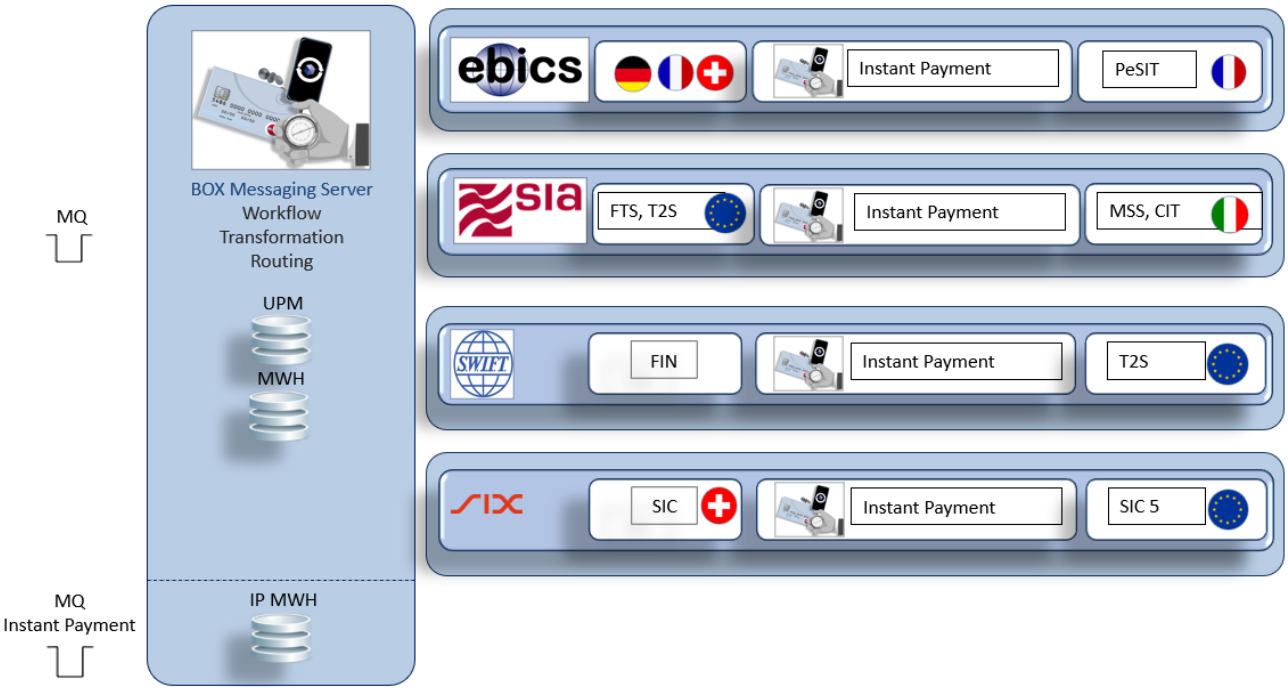
Picture 1 SEPA Instant Credit Transfer /SCT Inst) Overview

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 Messaging Hub

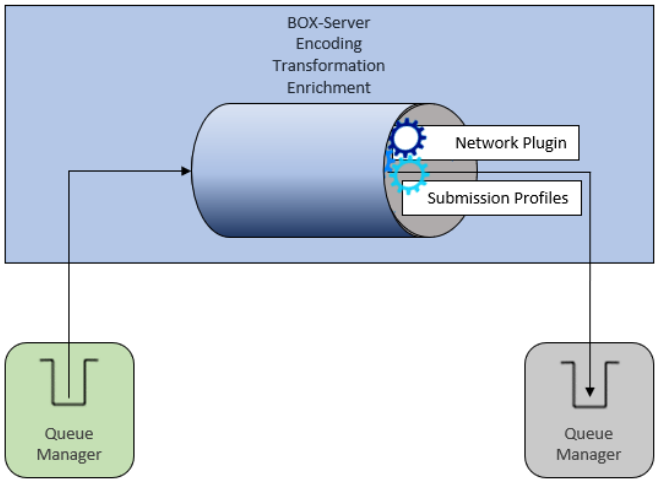
BOX represents a financial gateway and Messaging Hub integrating with back-office systems and multiple networks as illustrated below in the graph. BOX is a multi-network solution. Within that, BOX is 'SWIFT Customer Security Programme (CSP)' certified and supports all Swift business areas and interfaces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



Picture 2 BOX Messaging Hub Messaging Overview

2.2 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, SIC, EBICS).



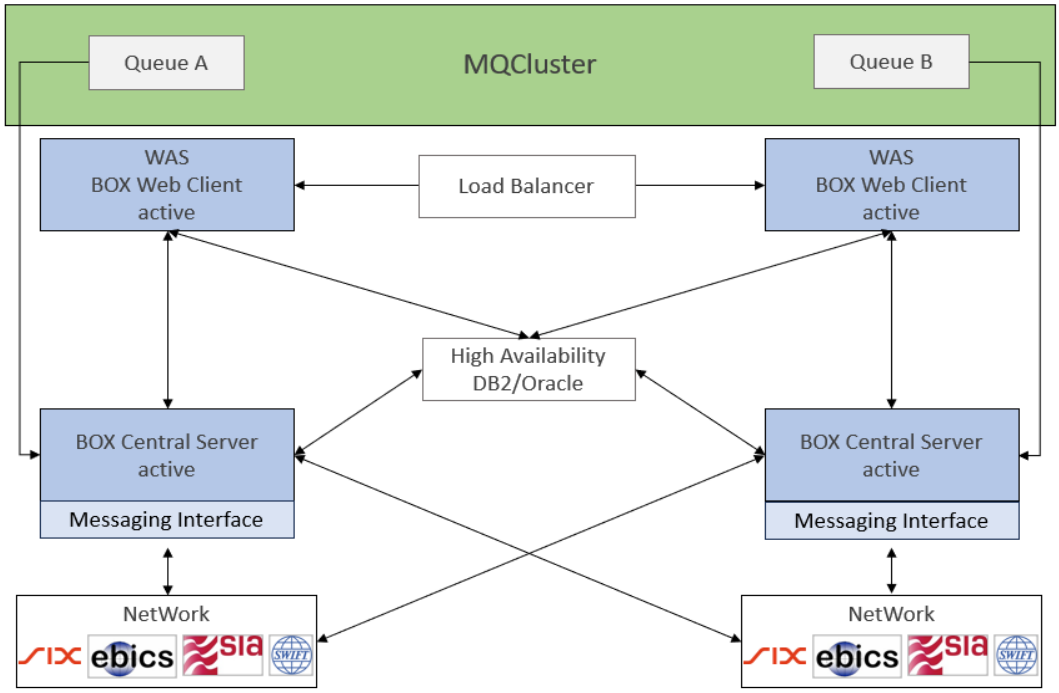
Picture 3 Instant Payments Message Processing

The 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.

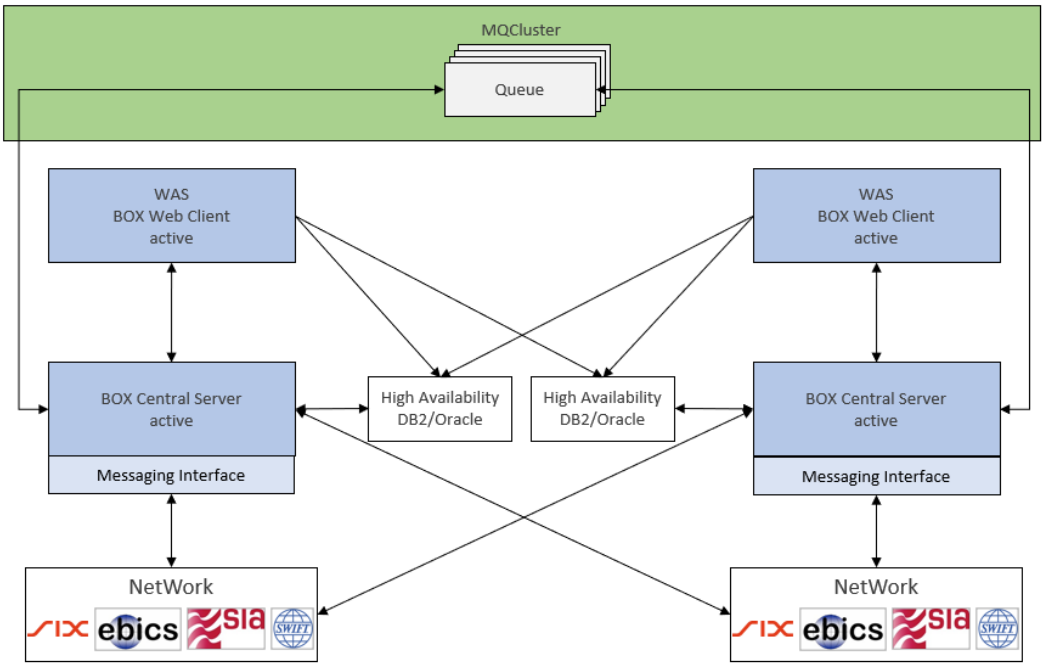
2.3 Active-Active

The IP System can be set up as in Active-Active System with a load balancer and a shared database.



Picture 4 Active-Active Overview with shared database

A further configuration results in two different systems, with one MQ cluster. Each system has its own database. Message data can be shared via the GUI. The other databases will be integrated via data sources in the web application server.

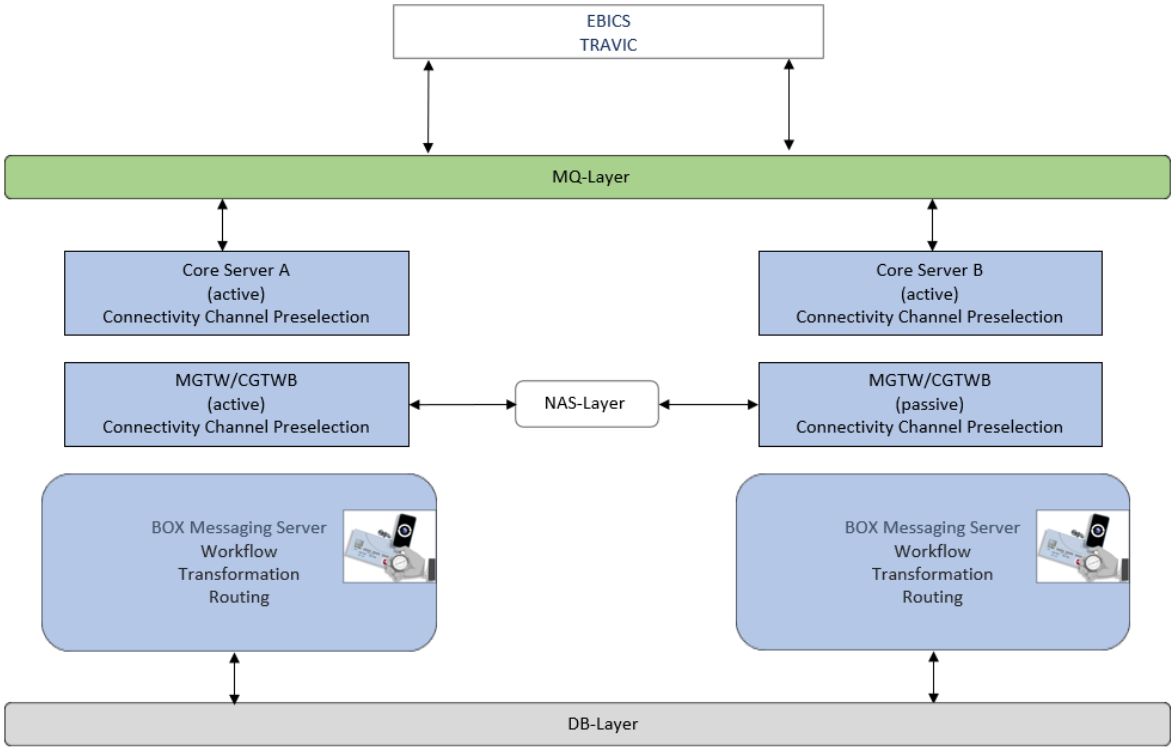


Picture 5 Active-Active Overview with database attached to each system

2.4 Multi Network Connectivity

2.4.1 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 5.7.3 and 5.8.1.



Picture 6 Exemplary EBICS Connection Overview

2.4.2 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 5.7.3.1 for a detailed description on the Alliance Gateway Instant (AGI) Plugin.

2.4.3 BOX connecting to SIA Net

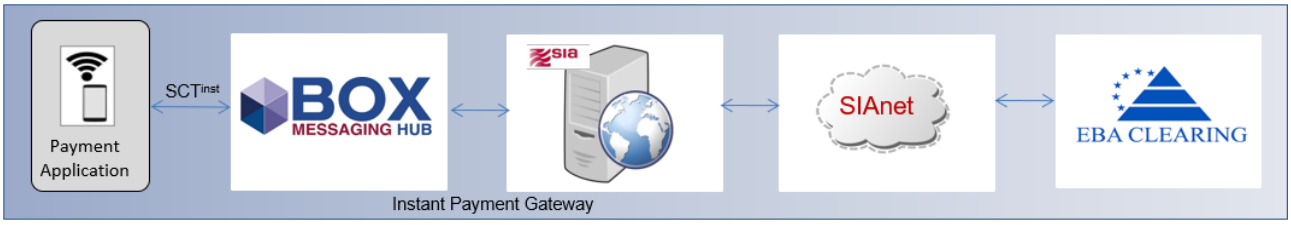
BOX connects to the SIAnet via a Messaging Integration exchanging Real-Time messages and files.

BOX offers the following integrations with SIAnet

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

Extending into Instant Payments

- SCT-Inst : TIPS, RT1

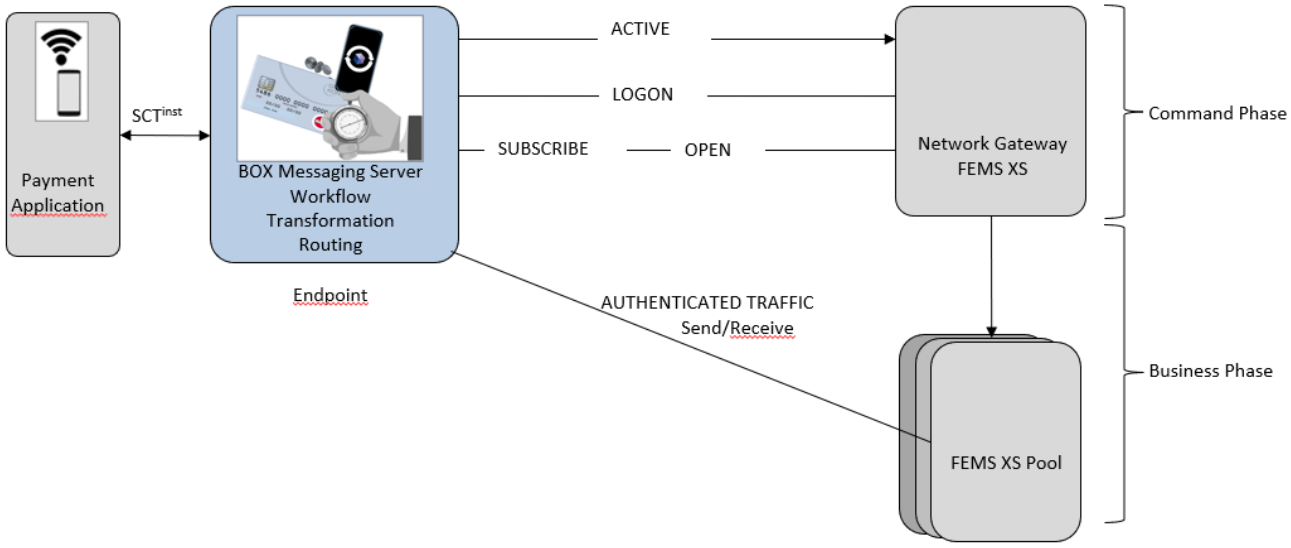


Picture 7 BOX Messaging Hub Connecting to SIAnet

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.

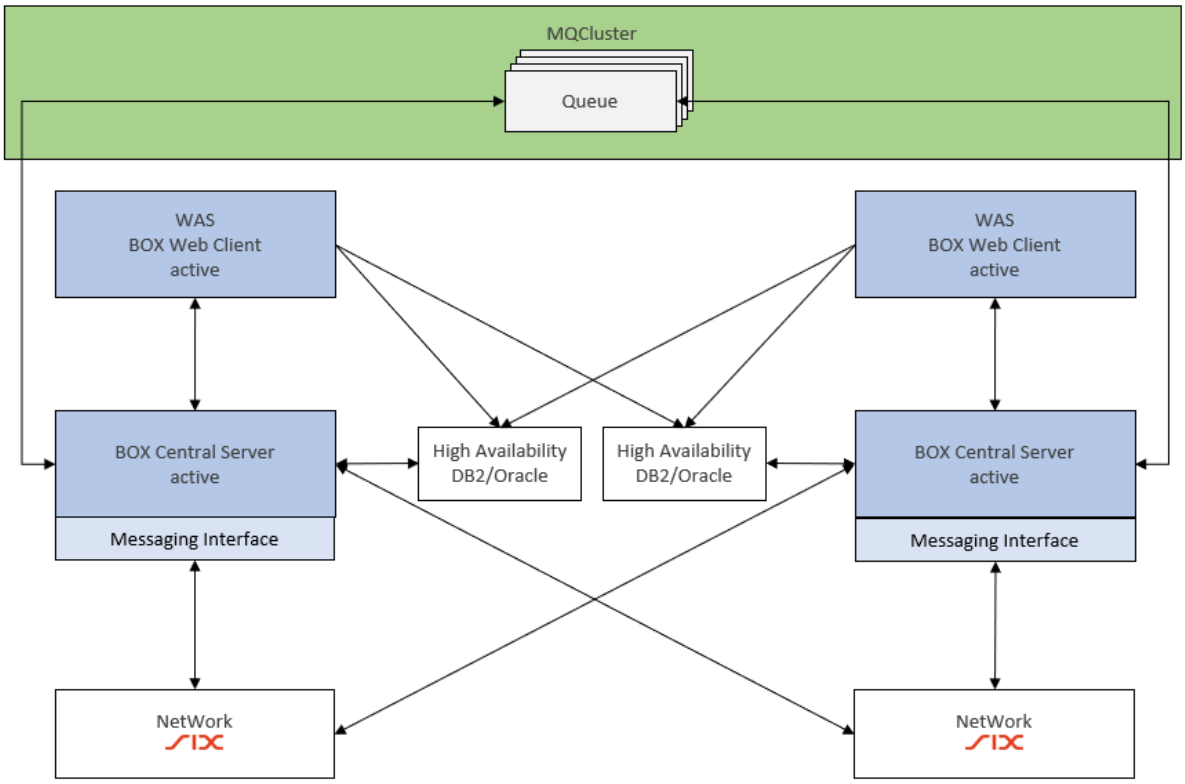
For Instant Payments, the Instant Message `eXtended` (IMX) Service provided by the FEMS XS is used. The following graph depicts the phases Command Phases, including the Logon and Subscription and the Business Phases Send and Receive.



Picture 8 Connecting to SIAnet: Command- and Business Phases

2.4.4 BOX connecting to SIX

The picture below shows an architecture, where two BOX instances use separate databases and one MQ cluster. A solution of SIX for a secure communication between SIX and business partners is the SIX Advanced Security Server (short SASS). The connectivity is established with the help of the SASS via Direct Connection P2P or the Secure Swiss Finance Network (SSFN). A specific configuration example can be found in chapter 5.10.2.



Picture 9 BOX Messaging Hub Connecting to SIX Overview

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

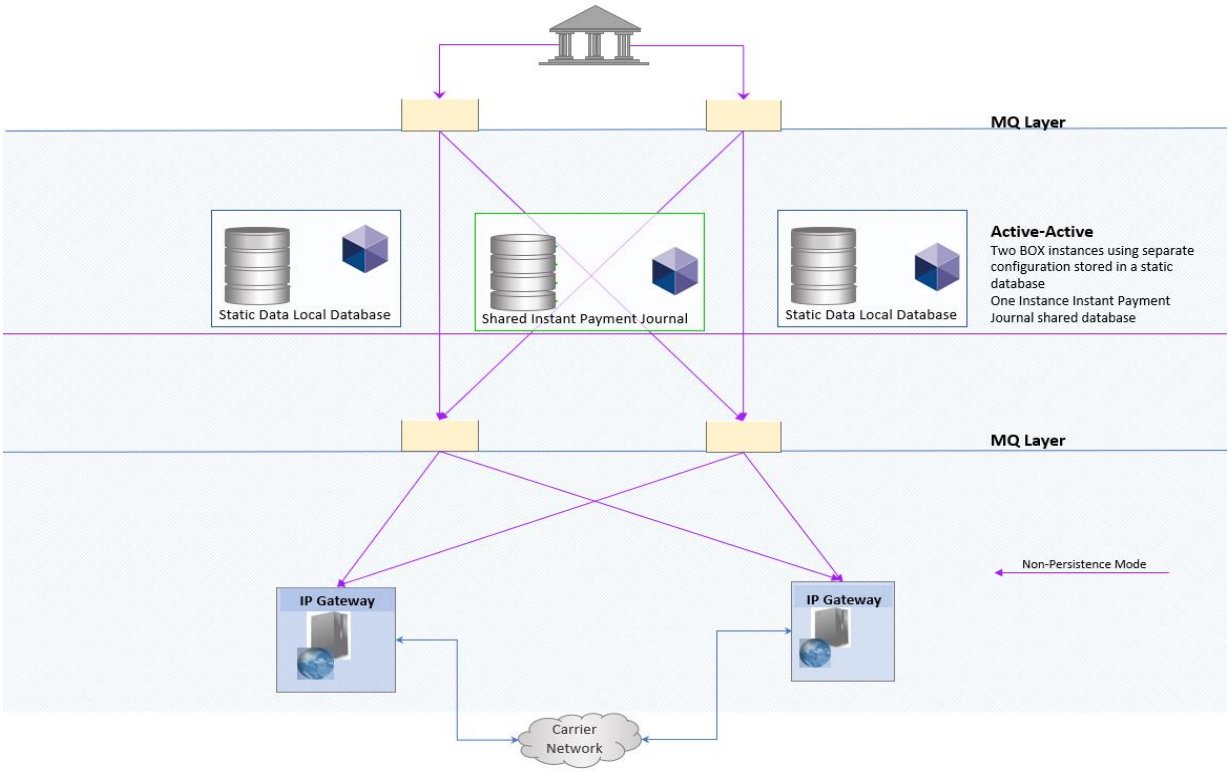
Value, String	Explanation
`InstPmtNoPersistence`	Instant Payment MPS using no persistence • Notification reconciliation data, Notifications create own MPS • Technical Acks • Can never change persistence level and should use only appropriate LCGs
`InstPmtJrnPersistence`	Instant Payment MPS using InstPmt 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

Table 2.4-I Persistence Modes

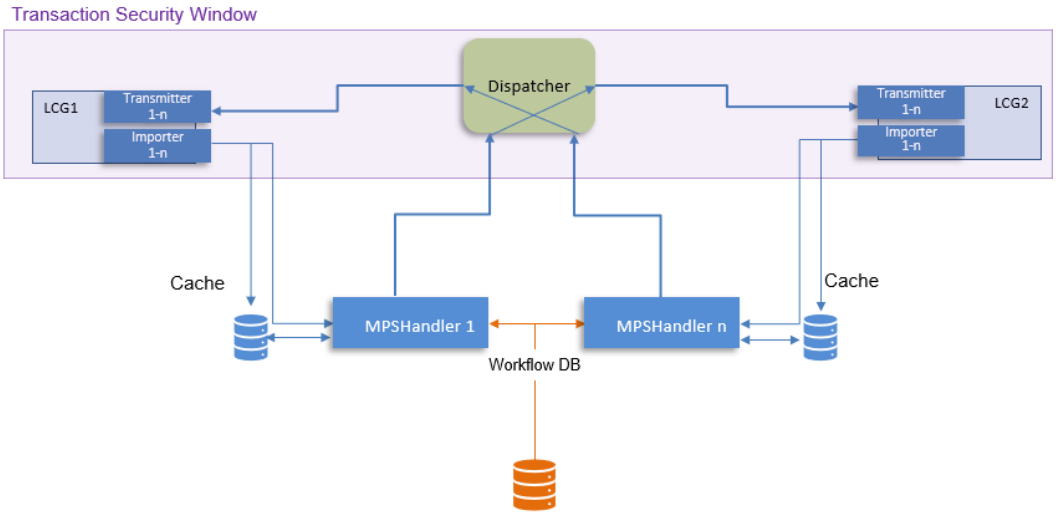
3.1 Operating BOX in Non-Persistence Mode (InstPmtNoPersistence)

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 has the processing of messages, which are not written to a database with all its attributes, but also the composition of messages to be delivered changed. It is important to understand the concept of message processing to avoid any configuration mistakes.



Picture 10 Non-Persistence Mode - Overview



Picture 11 Non-Persistence Mode – Configuration

Dispatcher

Collects 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 it is rolled back.

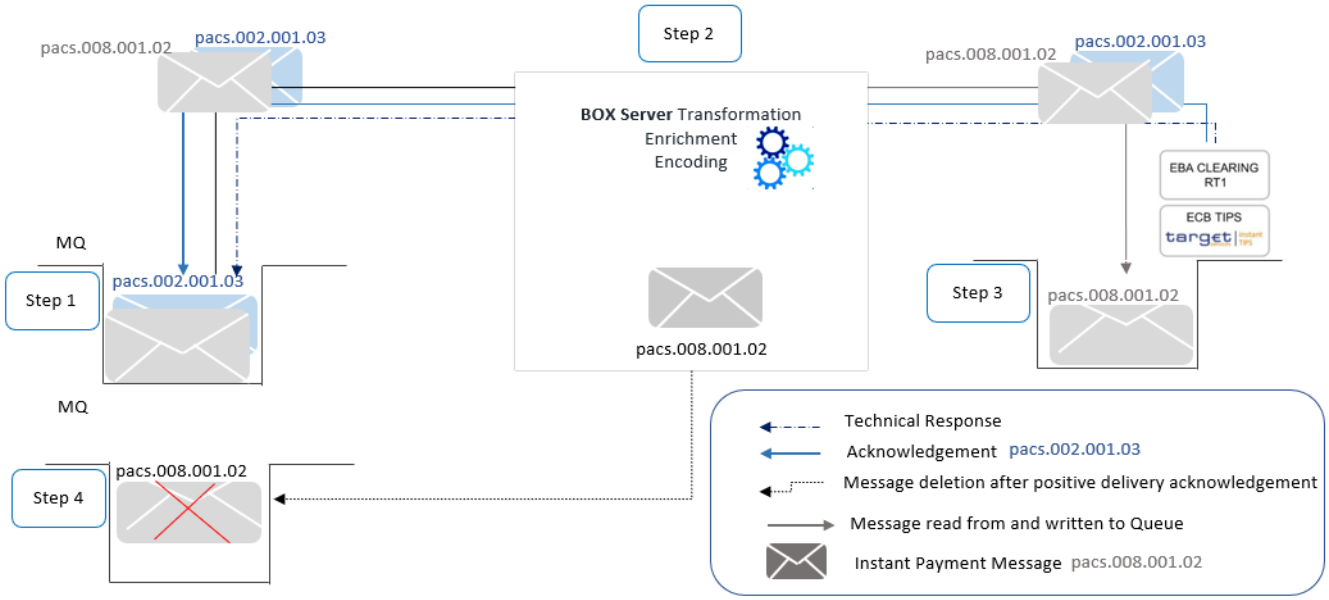
MPS Handler

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

3.1.1 Message Processing

The Instant Payments workflow can be roughly divided 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 0.



Picture 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-II 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. 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-IIIParameter 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-IV `DEFAULT_DELIVERY_COMPOSITION = 0x0010101`

IMPORTANT

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

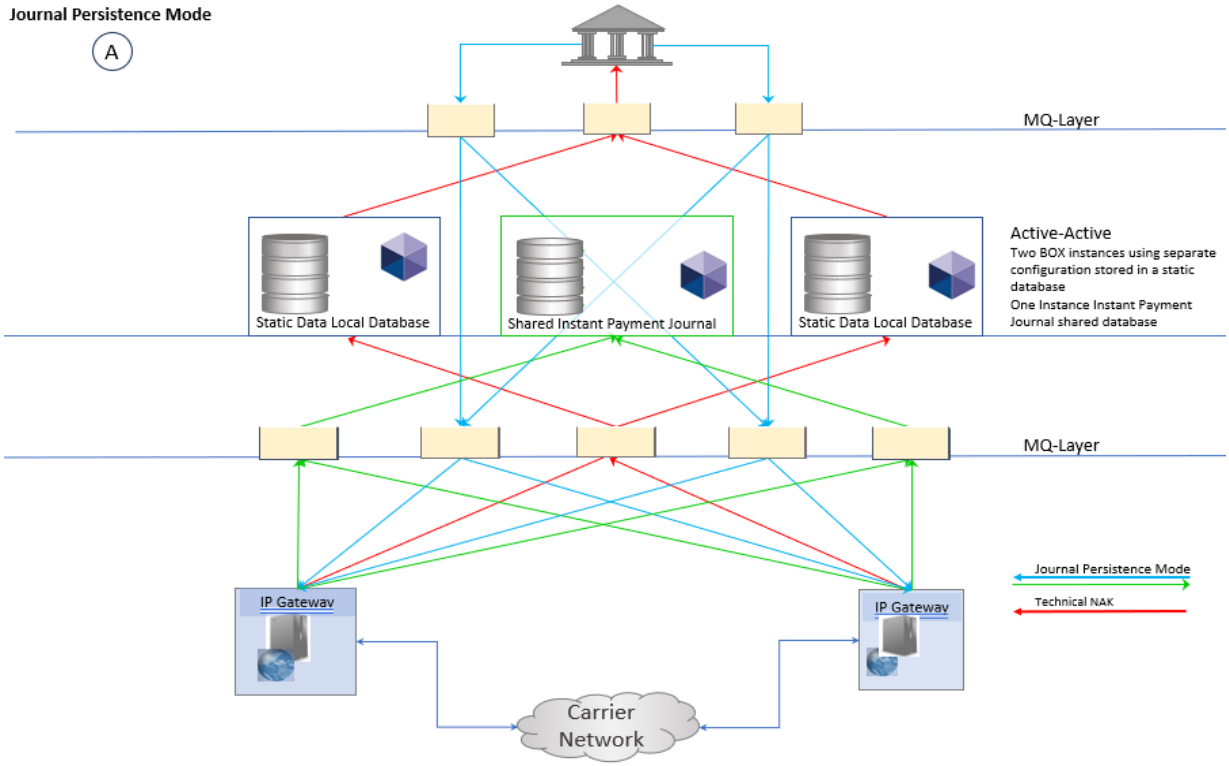
The Non-Persistence delivery composition and its possible values to be configured in the channel's configuration or workflow 'Delivery Instruction':

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

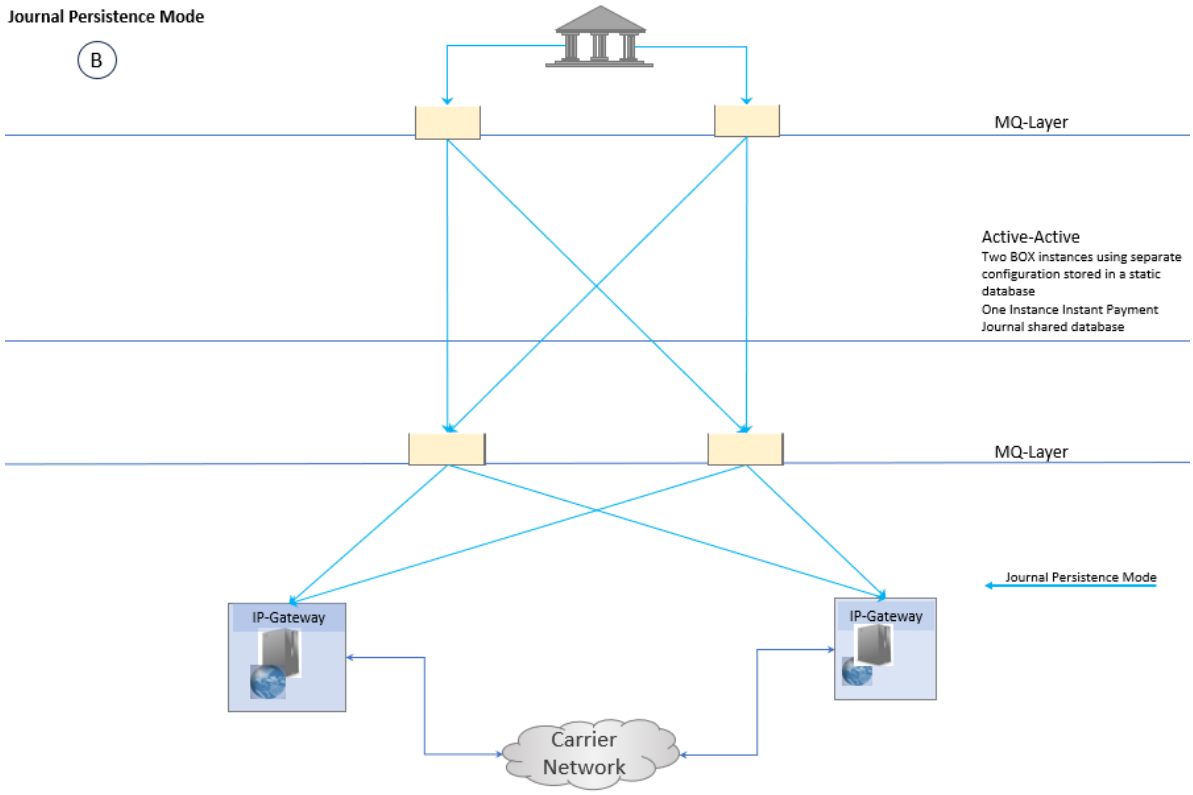
Table 3.1-V `DEFAULT_DELIVERY_COMPOSITION = 0x0000010`

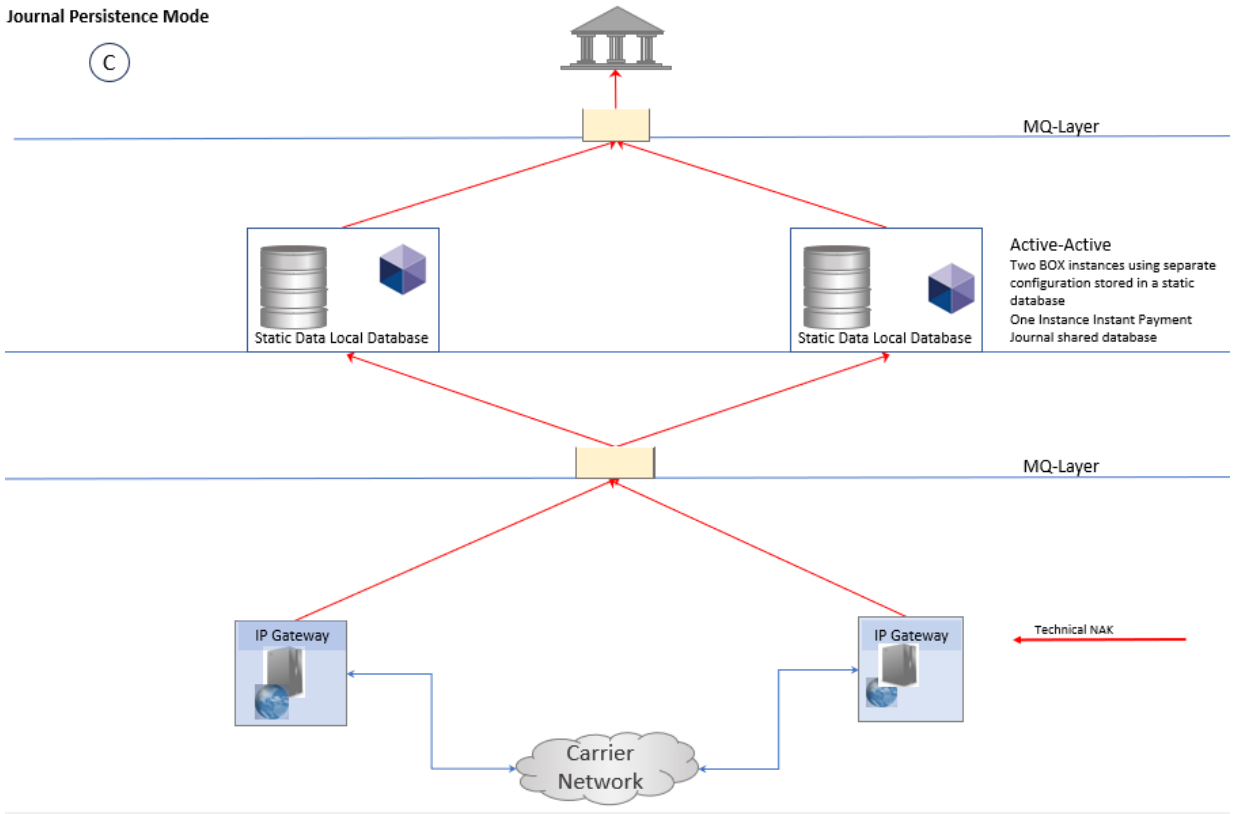
3.2 Operating BOX in Instant Payment Journal Persistence (InstPmtJrnPersistence)

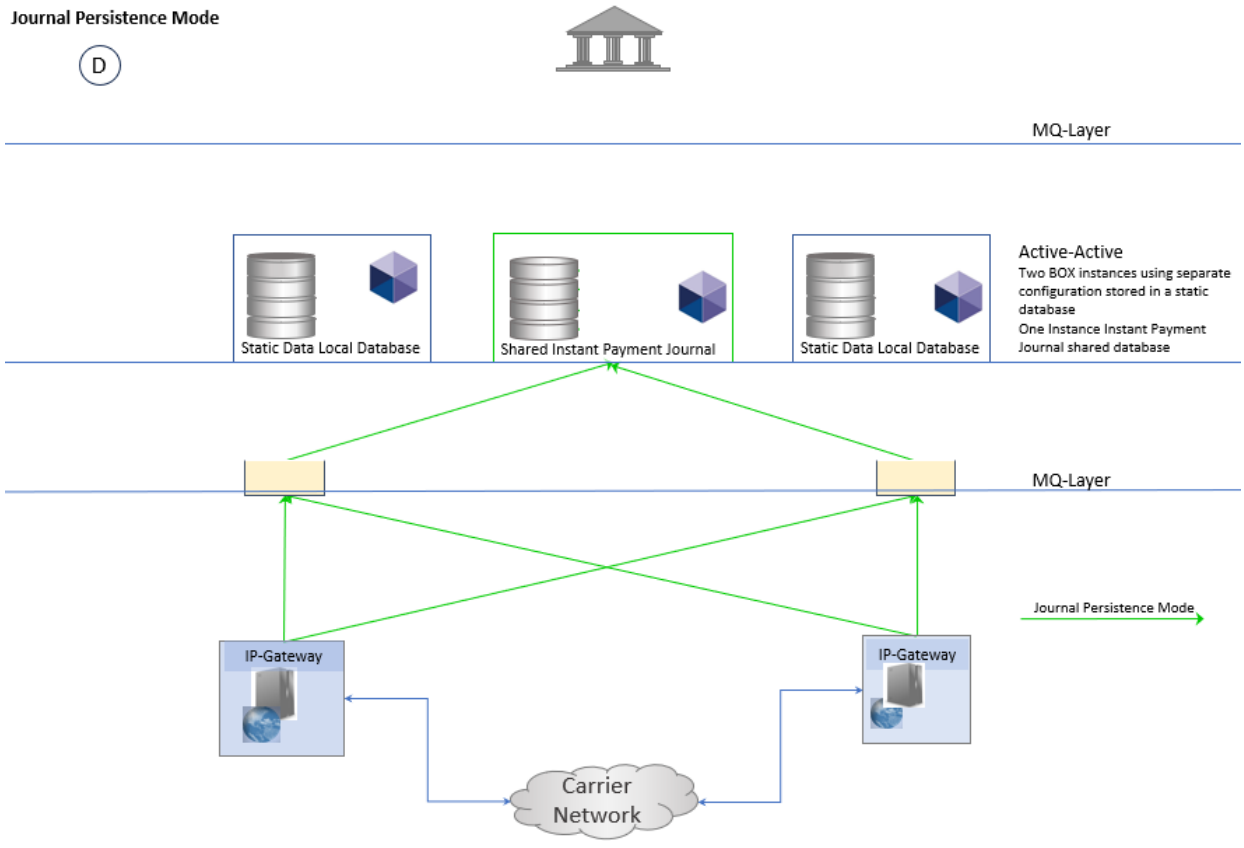
The ‘Instant Payment Journal Persistence’ mode refers to Instant Payment messages being written to a shared Instant Payment Database using the Instant Payment-Database `ProcessingSequenceID` allocation. It is possible to enhance the persistence level to level MP_MPS_PERSISTENCE_IP_FULL. The transaction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

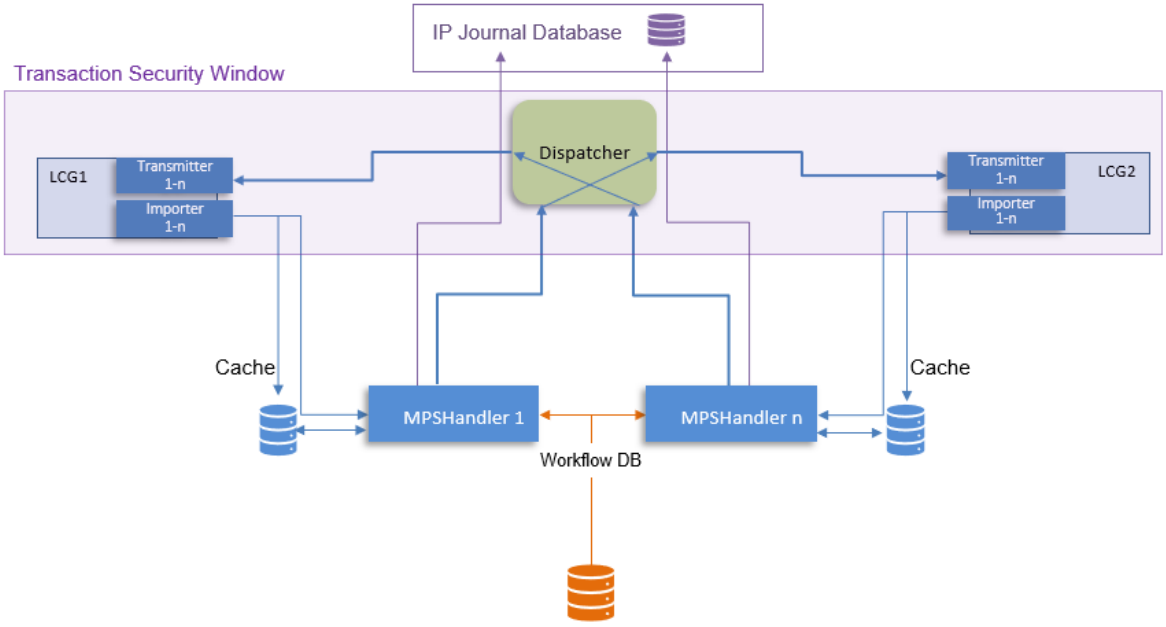


Picture 13 Instance Payment Journal Persistence Mode - Overview




Journal Persistence Mode
C
MQ-Layer
Static Data Local Database
Static Data Local Database
Active-Active
Two BOX instances using separate configuration stored in a static database
One Instance Instant Payment Journal shared database
MQ-Layer
IP Gateway
IP Gateway
Technical NAK
Carrier Network

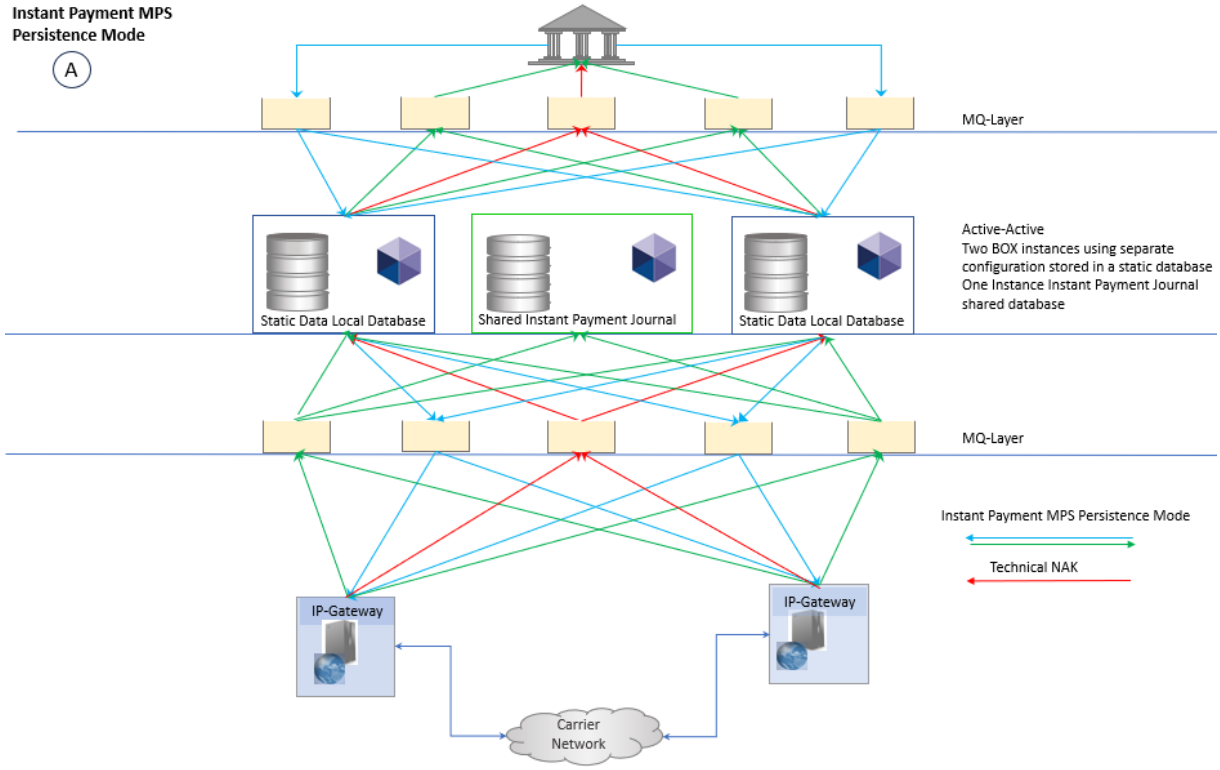

Journal Persistence Mode
D
MQ-Layer
Static Data Local Database
Shared Instant Payment Journal
Static Data Local Database
Active-Active
Two BOX instances using separate configuration stored in a static database
One Instance Instant Payment Journal shared database
MQ-Layer
Journal Persistence Mode
IP-Gateway
IP-Gateway
Carrier Network



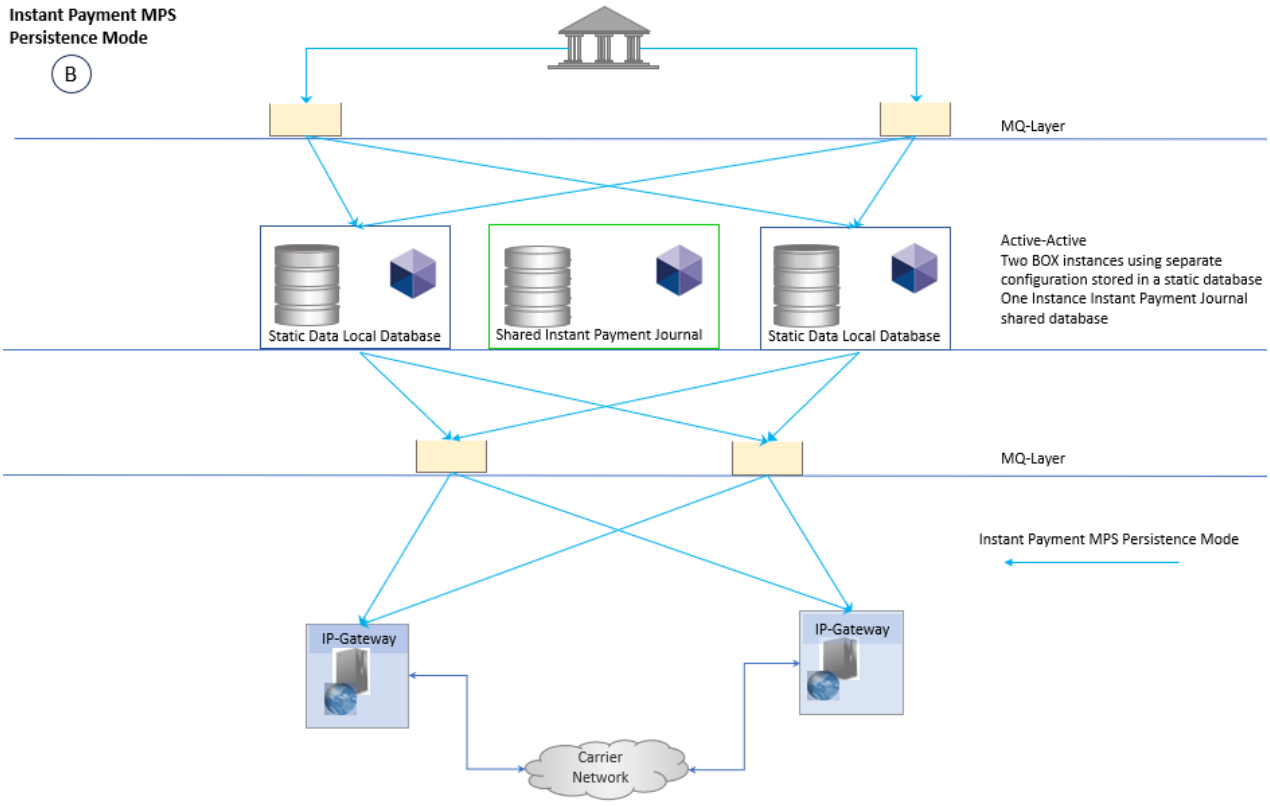
Picture 14 Instance Payment Journal Persistence Mode – Configuration

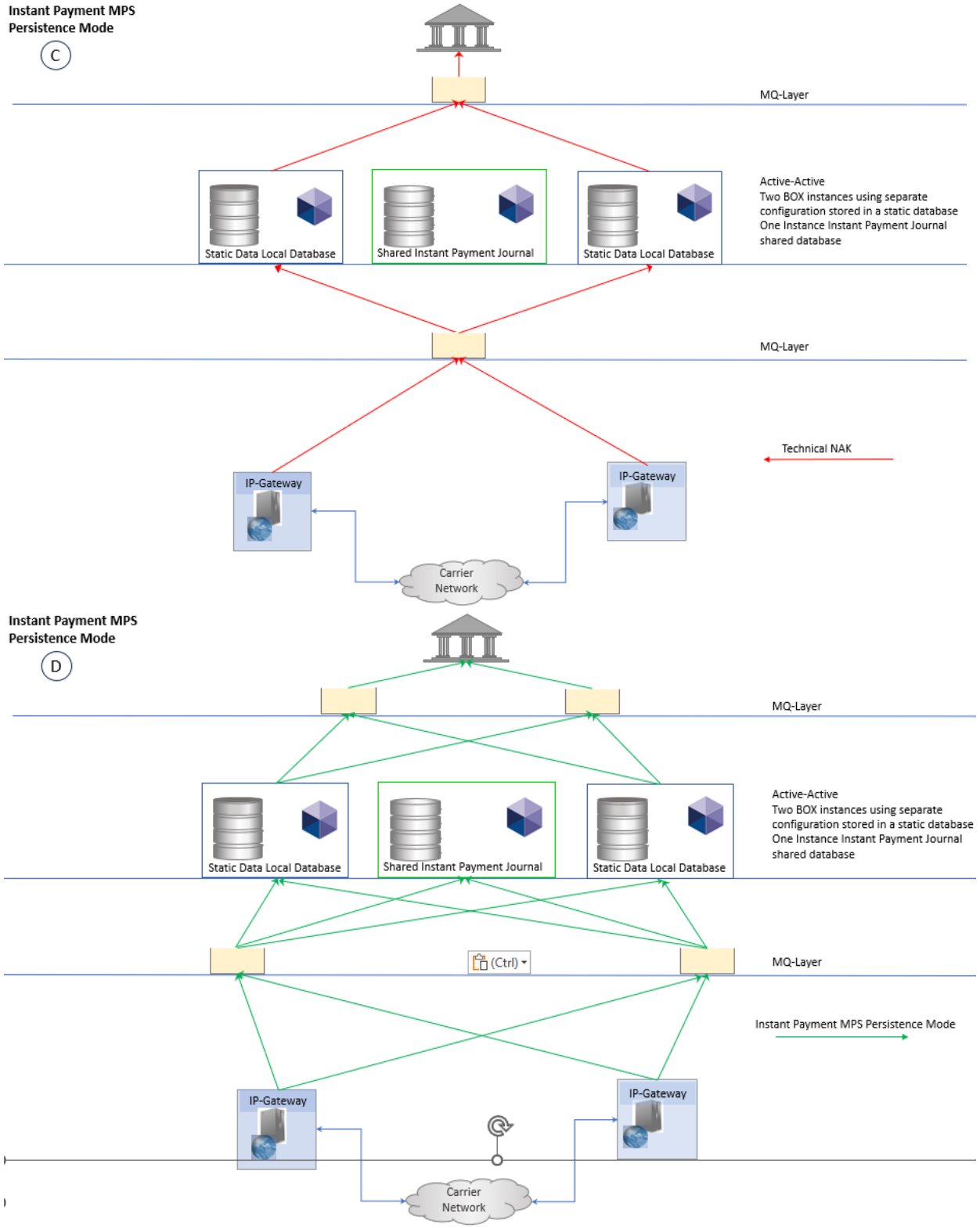
3.3 Operating BOX in Message Processing Sequence (MPS) Persistence (InstPmtMPSPersistence)

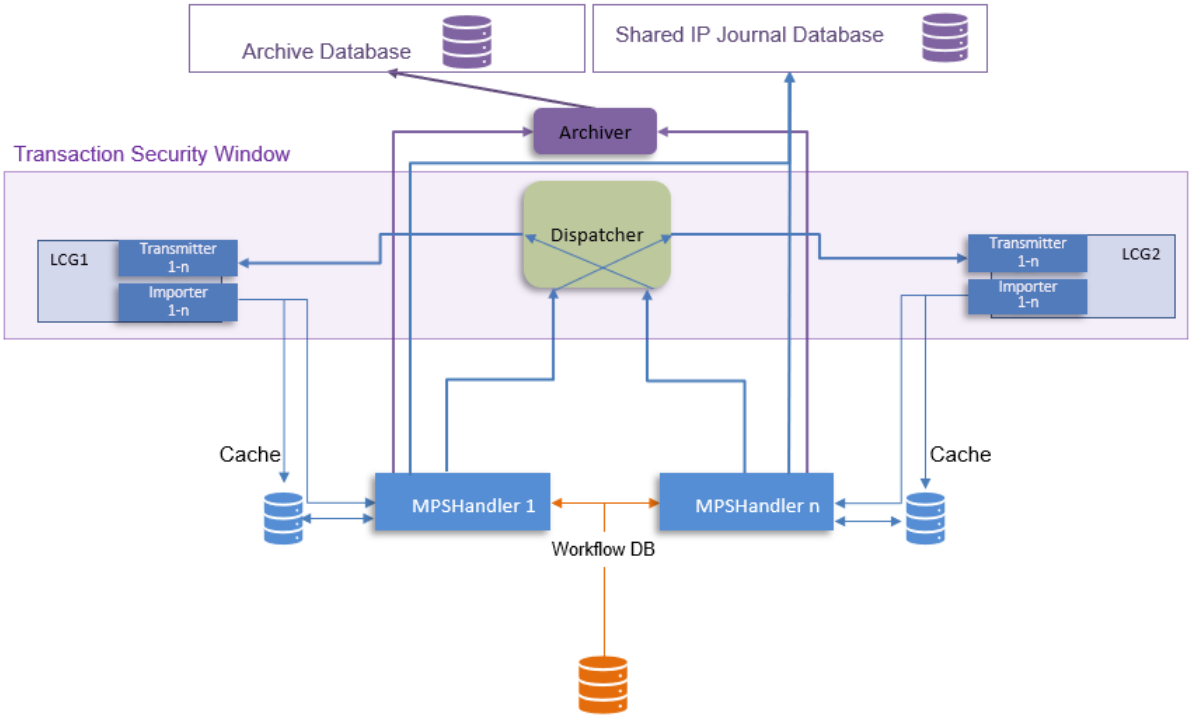
Instant Payment MPS is written to the Instant Payment - Journal in a shared IP-database and written to MPS tables in the local database, using the Instant Payment-Database `ProcessingSequenceID` allocation traditional transaction handling based on MPS instruction persistence in the database.



Picture 15 Instant Payment MPS Persistence Mode – Overview



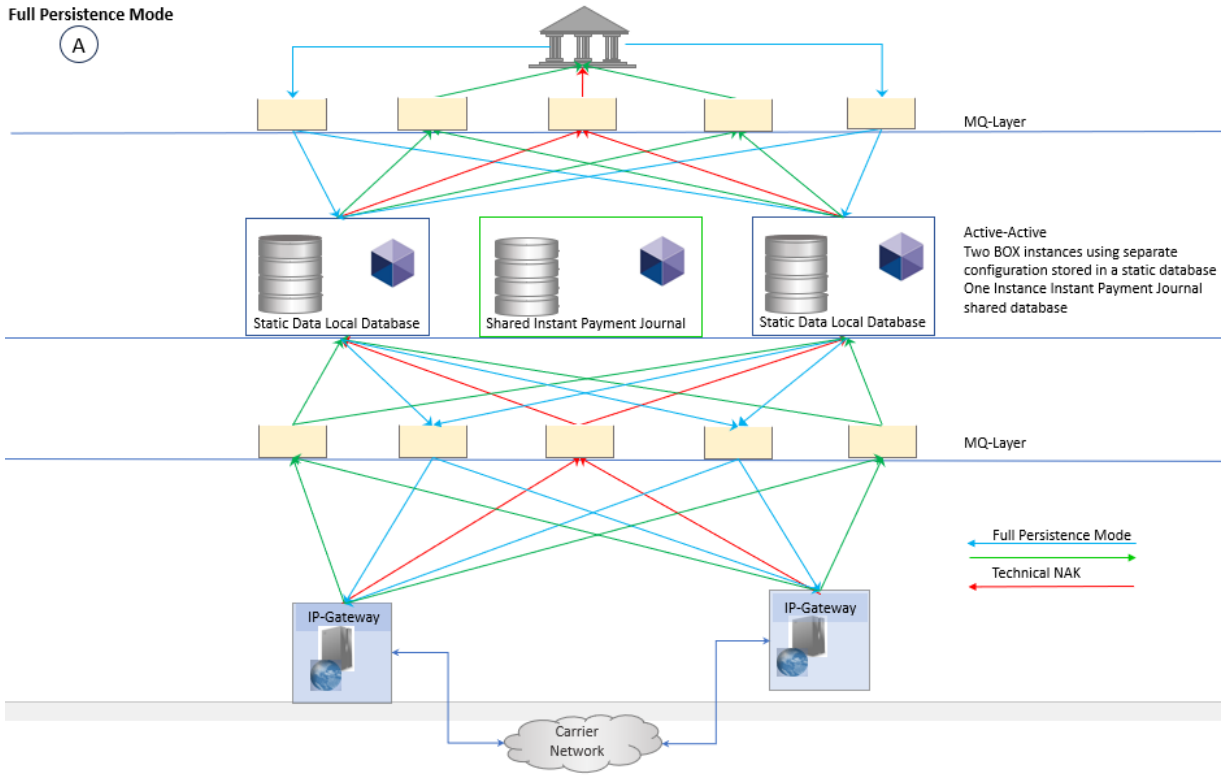

Instant Payment MPS
Persistence Mode
C
MQ-Layer
Static Data Local Database
Shared Instant Payment Journal
Static Data Local Database
Active-Active
Two BOX instances using separate configuration stored in a static database
One Instance Instant Payment Journal shared database
MQ-Layer
Technical NAK
IP-Gateway
IP-Gateway
Carrier Network
Instant Payment MPS
Persistence Mode
D
MQ-Layer
Static Data Local Database
Shared Instant Payment Journal
Static Data Local Database
Active-Active
Two BOX instances using separate configuration stored in a static database
One Instance Instant Payment Journal shared database
(Ctrl)
MQ-Layer
Instant Payment MPS Persistence Mode
IP-Gateway
IP-Gateway
Carrier Network



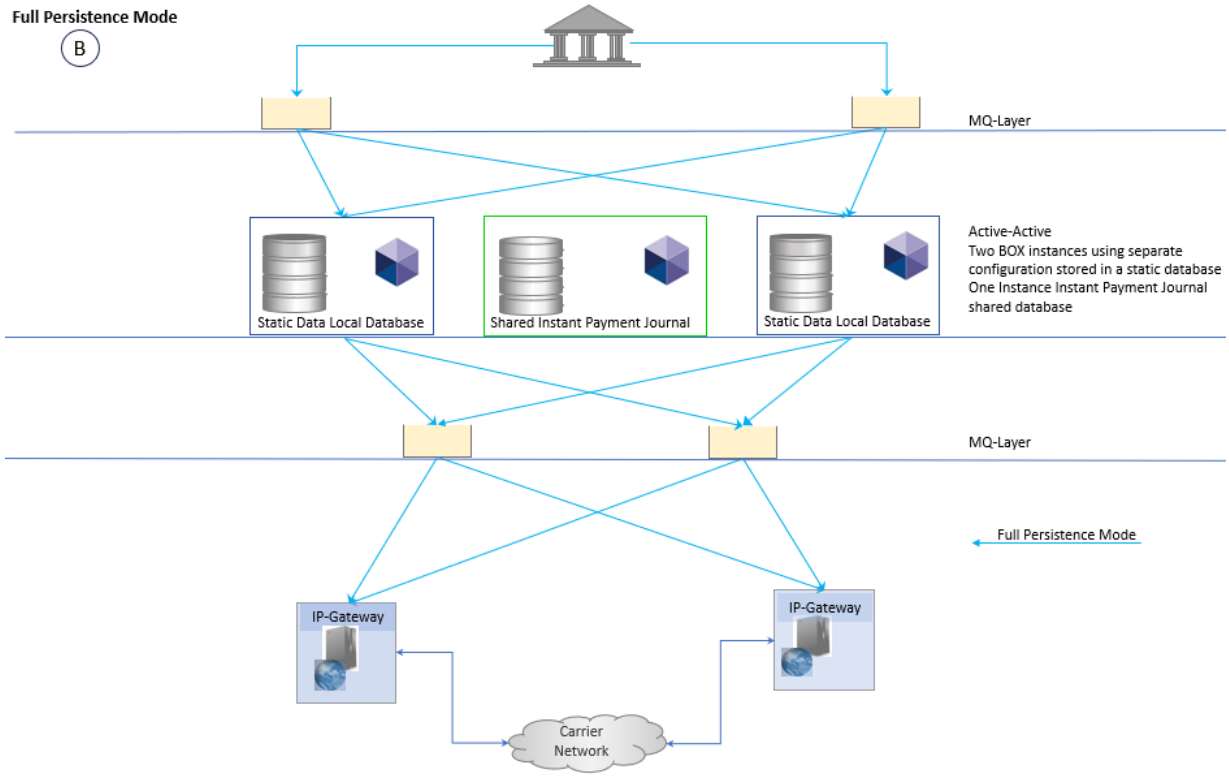
Picture 16 Instant Payment MPS Persistence Mode - Configuration

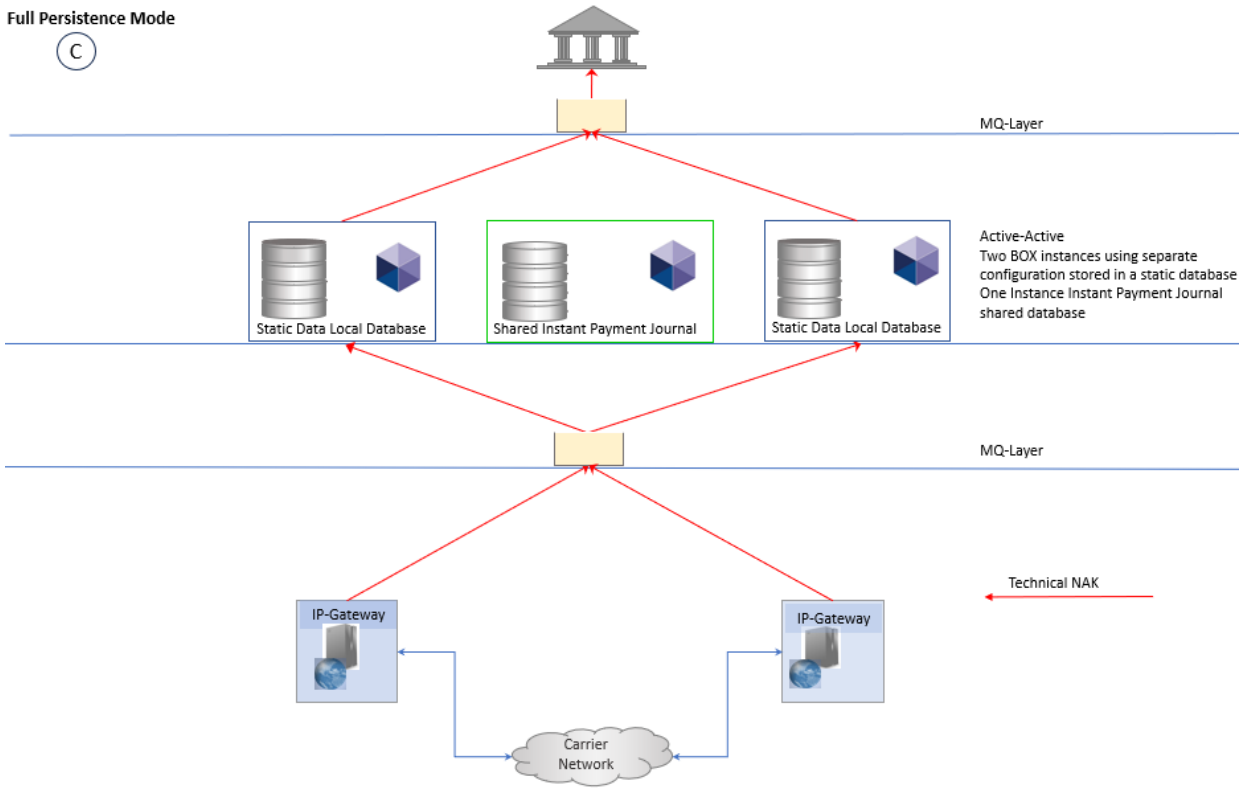
3.4 Operating BOX in Full Persistence Mode

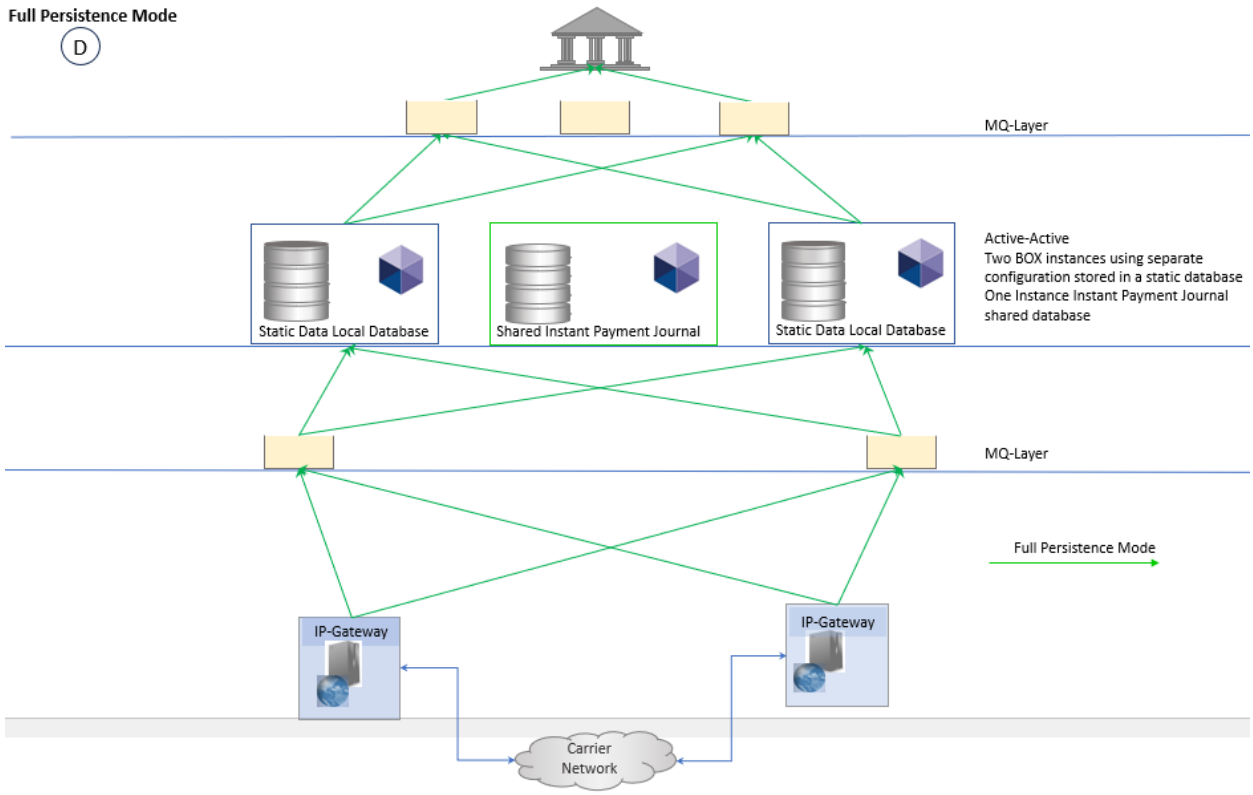
The messages are written to a message warehouse in the local database, using the traditional transaction handling based on MPS instruction persistence in the database.

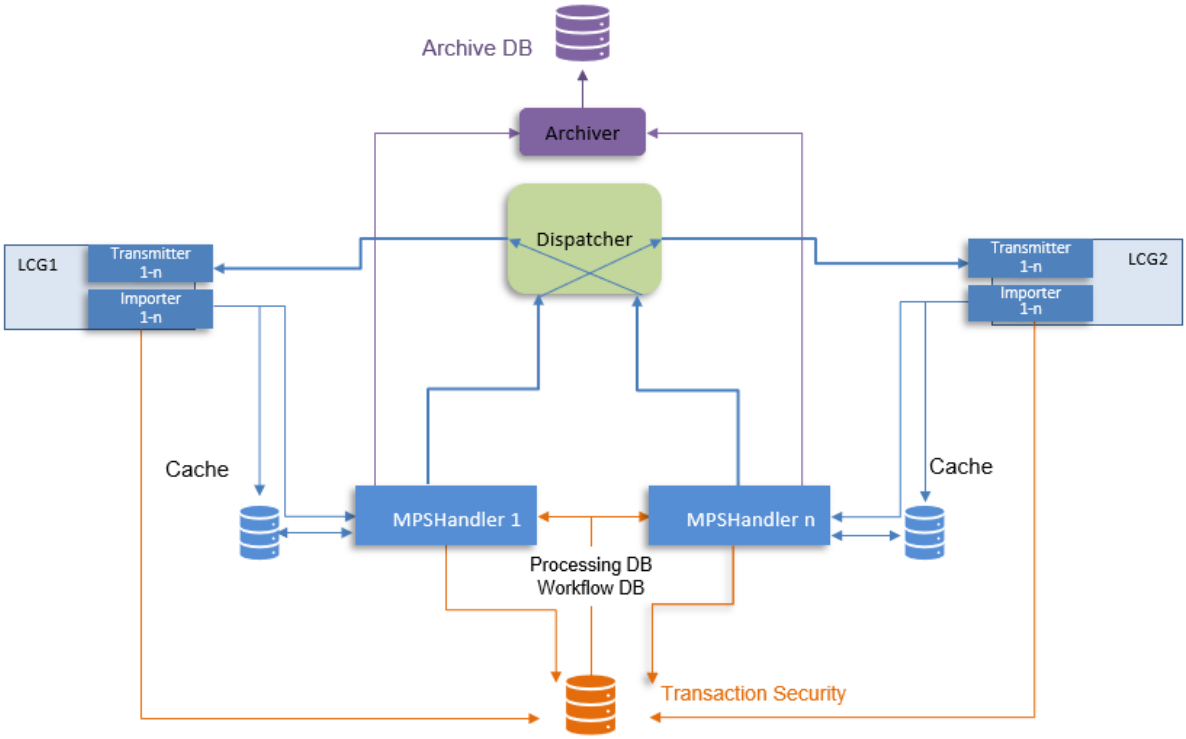


Picture 17 Full Persistence Mode – Overview




Full Persistence Mode
C
MQ-Layer
Static Data Local Database
Shared Instant Payment Journal
Static Data Local Database
Active-Active
Two BOX instances using separate configuration stored in a static database
One Instance Instant Payment Journal shared database
MQ-Layer
Technical NAK
IP-Gateway
IP-Gateway
Carrier Network


Full Persistence Mode
D
MQ-Layer
Static Data Local Database
Shared Instant Payment Journal
Static Data Local Database
Active-Active
Two BOX instances using separate configuration stored in a static database
One Instance Instant Payment Journal shared database
MQ-Layer
Full Persistence Mode
IP-Gateway
IP-Gateway
Carrier Network



Picture 18 Full Persistence Mode – Configuration

Dispatcher

Collects 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

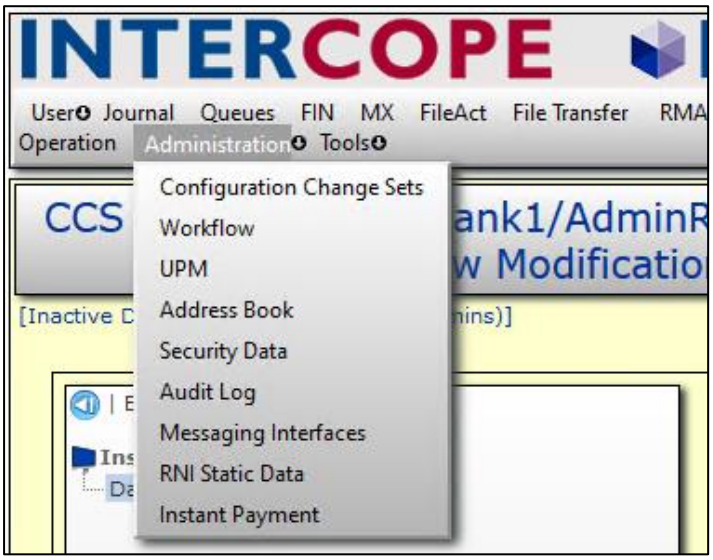
In case of technical errors, an NAK results in the generation of an MPS, which is written to a dedicated error queue and consequently dealt with according to the predetermined error handling on customer side.

3.5 BOX Instant Payments (IP) Journal Archive

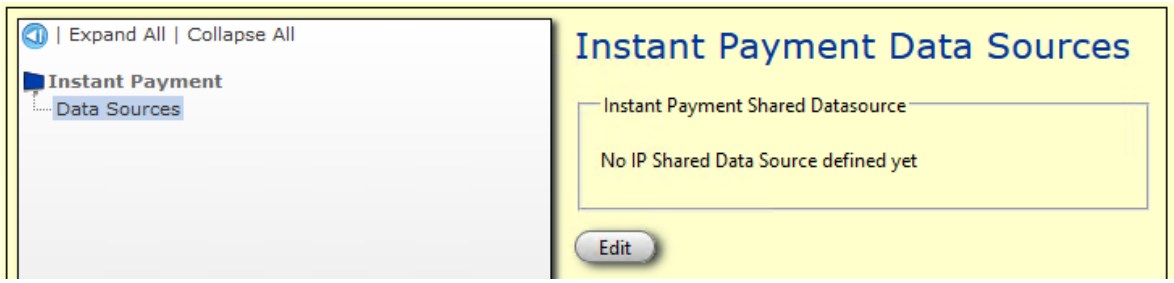
To set up the IP Journal Archive, create as a first step the IP Data Source.

1. Exemplary Data Source

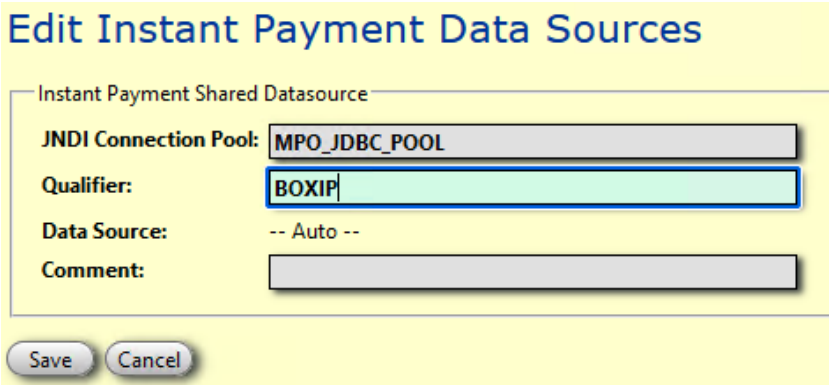
- Open Instant Payment from Administration



- Edit



- Enter the respective JNDI Connection, Qualifier and, if applicable a Comment



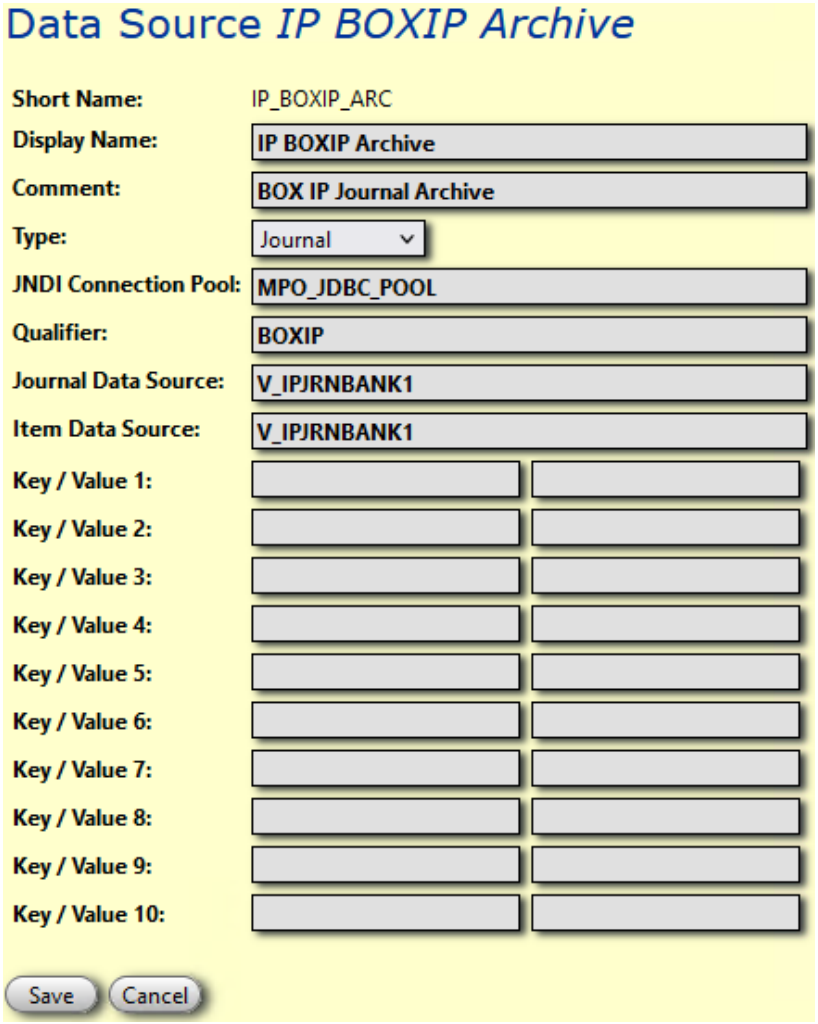
- Save

2. Create Data Source for Workflow

To create a new application queue definition, go to Administration -> Workflow ->Data Source.

Please note, it is also possible to s

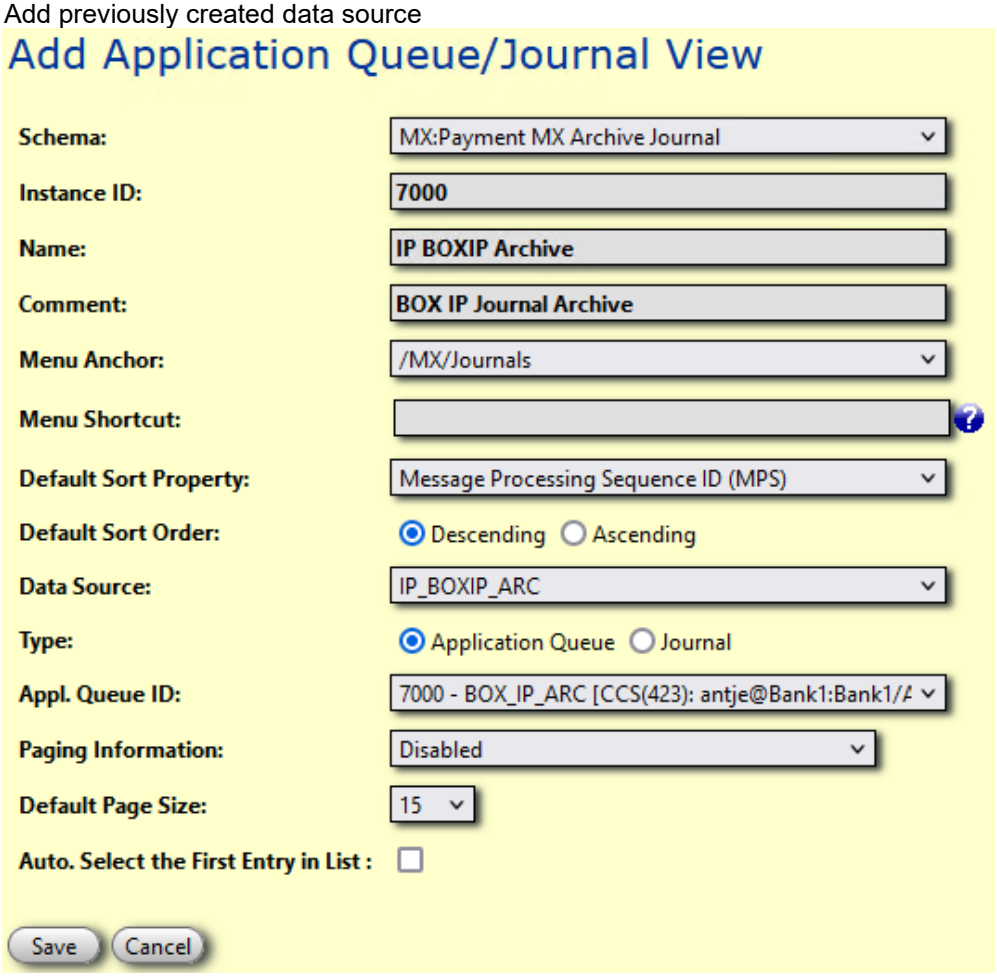
- Create a Data Source of type ‘Journal’ with the created JNDI Connection Pool.



3. Exemplary Application Queue Definition

To create a new application queue definition, go to Administration -> Workflow -> Journals/Queues -> Application Queue Definitions.

- Create an Application Queue Definition
- Create Views&Tasks -> Application Queue/Journal View with Schema MX: IPRT-Archive-Journal
- Add necessary parameters
- Add previously created data source



4 Database Creation for Instant Payments

The Instant Payment (IP) Database is separated from and smaller in size than the BOX database. To install it, BOX offers the database tool dbxml, which includes all necessary executables and structure to create the IP database. Once installed, the database provides for a journal view.

Please refer to the documentation box_dbtool_v<x>.pdf for detailed information regarding the tool and the creation of the IP database.

Depending on the product to be installed, the tool's configuration file needs to be adapted.

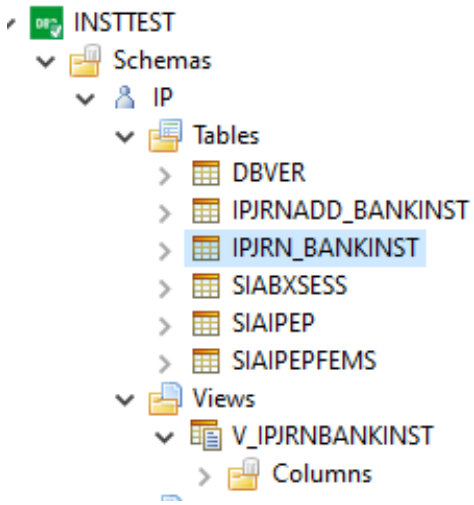
Within the configuration file, please adjust the parameter product:

```
<product>$$R$PRODUCT$</product>
```

Instant Payment related tables are installed, when setting one or more (depending on product) of the following:

```
"BOX_SIA"> SIA specific tables
"BOX_IP"> BOX related tables for Instant Payment
"IP"> Further BOX related tables for Instant Payment
"MPO_SHARED"> MPO Shared Tables needed e.g., for SIC
"PESIT"> PESIT related tables
"EBICS_IB"> EBICS Interbank related tables
"SIC"> SIC related tables
```

Example of creating and installing 'IP' related tables and views:



Picture 19 Instant Payment Database Overview

Please note, BANKINST reflects the company name.

5 Instant Payment Configuration Specifics

5.1 Submission Profiles

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

Available Profiles:

- Interact
- FileAct – Put File
- FileAct – Get File
- SIA FLS – Send File
- SIA FTS – Send File
- SIA T2S – Send Message/File
- SIA Send Instant Payments Realtime Message (Example chapter 5.9.1)
- EBICS – Send Instant Payments Realtime Message (Example chapter 5.8.3)
- EBICS – Interbank File Transfer
- SWIFT – Send Instant Payments Realtime Message (Example chapter 5.7.4)
- SIC - Send Message
- SIC IP - Send Message (Example chapter 5.10.6)

5.2 Clearing and Settlement Mechanisms

BOX supports the following CSMs:

- RT1
- TIPS
- RT1/TIPS Instruction Party

5.3 Instant Payment Message Journal

The Message Journal dedicated to Instant Payment (IP) contains separate IP ISO20022 messages (payments) for a relatively brief 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 an input message with a Technical Ack/Nak and the Non-Repudiation Data.. Also, IP ISO20022 messages are reconciled to form an SCTinst payment.

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

5.3.1 Data Sources

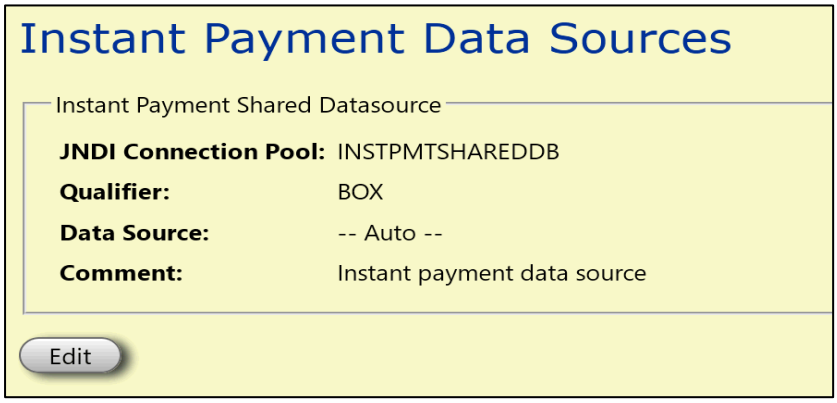
Two options are available when configuring the Instant Payment Journal's data sources, i.e. setting up database definitions for such journals:

5.3.1.1 Single Data Source

A single (logical) database is used, which is accessed by all BOX clients and servers.

The configuration in the Client's GUI of this data source is done through the menu item

`Administration / Instant Payments :`



Picture 20 Instant Payment Journal with Predefined single Data Source

The configuration for the server is done through the section `[SHARED_DB_INTERFACE]` in the Central Server Module (and Messaging Gateway Modules hosting SIA FEMS channels).

In this database the following objects must exists:

→ For each (BOX-)Company, tables named
`IPJRN_<ClientPrefix>`
`IPJRNADD_<ClientPrefix>`

and views named
`V_IPJRN<ClientPrefix>`

Please note, it is recommended that the 'configuration' database (section `[DB_INTERFACE]`) should be set to the same database. Different central server - and messaging gateway modules must use different Module Ids! (Stateful Active-Active cannot be used with Instant Payments!)

5.3.1.2 Multiple (independent) Data Sources - Silo Architecture

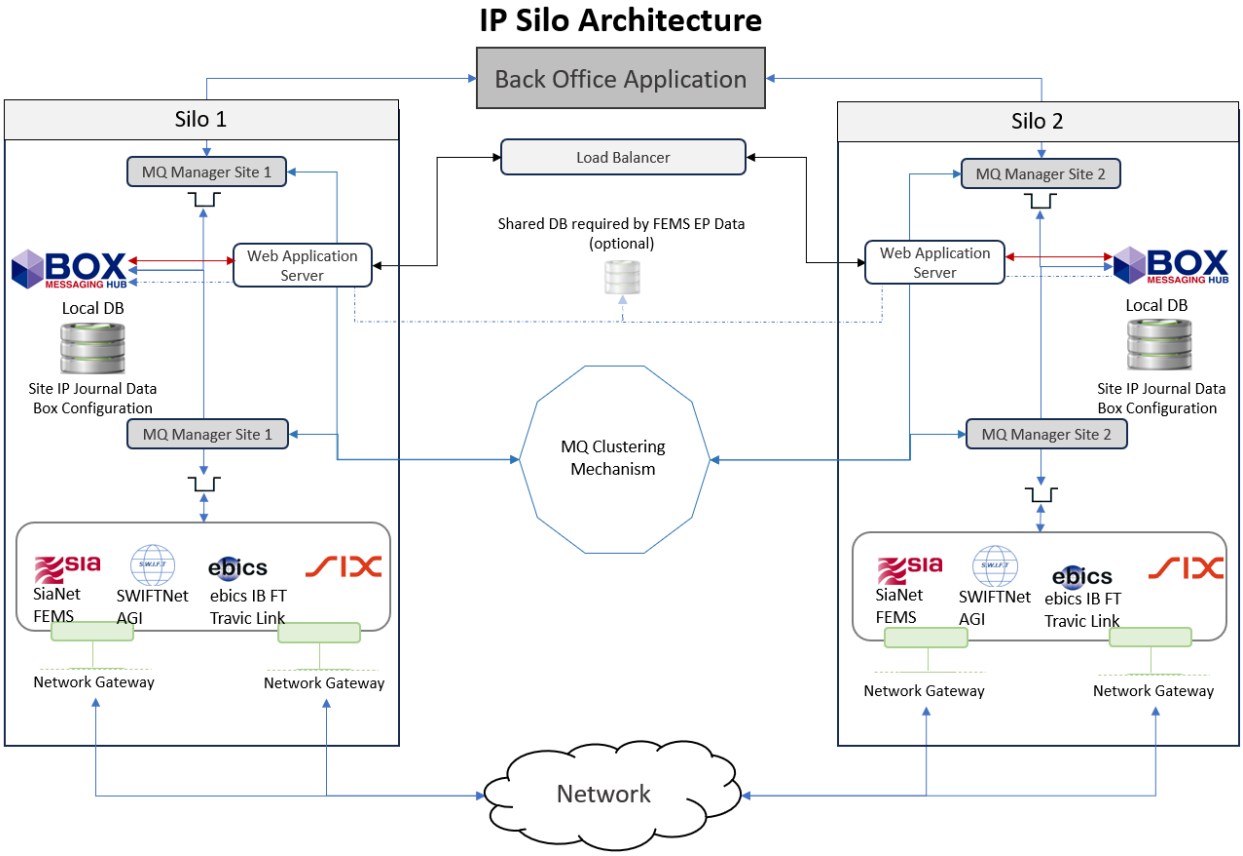
Several independent databases are used, which are located and hosted in different data centers (locations) and accessed only by 'local' BOX Server Modules to store `Instant Payment Journal` data. Between these databases a connection is not required, and BOX server modules only need to connect to the database in 'their' silo.

Combined View on all databases giving access to the BOX Client's GUI

The BOX Client's GUI offers a combined view into these different databases, i.e., the GUI requires access to these different databases.

It is still assumed that esp. the UPM setup is a logically single UPM, i.e., all databases contain the same UPM configuration (Objects, Names, Client Prefixes). Usually, the other configuration would also be similar in all databases (locations).

Only the Ids would differ:
Different central servers and messaging gateway modules must use different Module Ids!
The following overview is created with 2 silos, more silos in that architecture are possible and



Picture 21 Instant Payment Message Journal with Predefined multiple Data Sources

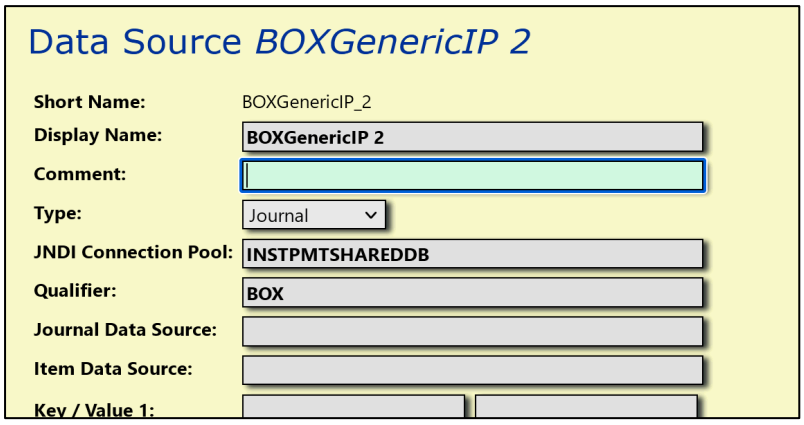
Stateful Active-Active cannot be used with Instant Payments!

Important

Please note, this setup is currently not supported with SIAnet FEMS, which requires a shared database.

The **BOX server configuration** is done through `[SHARED_DB_INTERFACE]` and `[DB_INTERFACE]` which should then point to the 'local' database within the specific location.

The **GUI configuration** must then configure one data source per location, which can be used for all <ClientPrefix>) through



Picture 22 Configuration of one Data Source per location

It is recommended, that

-> All UPM configurations in a distributed database configuration share the same <ClientPrefix>-values

In this case leave field `'Journal Data Source'` in 'Data Sources' used by `'Multiple Datasource'` Instant Payment Journals empty.

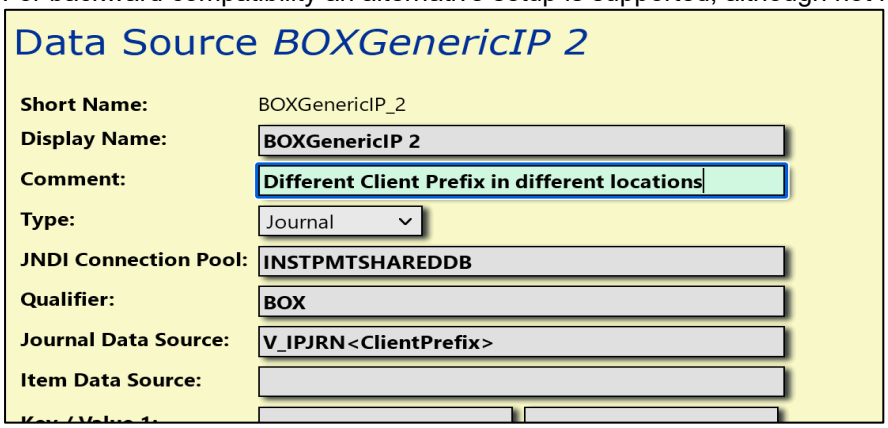
-> In each database the following objects must exist:

For each (BOX-)Company, tables named
- → `IPJRN_<ClientPrefix>` and
- → `IPJRNADD_<ClientPrefix>`

and views named
- → `V_IPJRN<ClientPrefix>`.

(All UPM configurations share the same set of `<ClientPrefix>`-values.)

For backward compatibility an alternative setup is supported, although not recommended:



Picture 23 Configuration with different Client Prefixes per location

Such a configuration must be done for all `<ClientPrefix>` used locally, and one such configuration can be used only with a single `<ClientPrefix>`.

In such a local database the following objects must exist:

For each specified `<ClientPrefix>` tables named
- → `IPJRN_<ClientPrefix>`

→ `IPJRNADD_<ClientPrefix>`

and views named

→ `V_IPJRN<ClientPrefix>.`

As a multi-data source Instant Payment Journal, the view uses multiple data sources. It is then possible to use different `ClientPrefix`-values for the 'same' BOX-Company in separate locations.

However, this is not necessary and not recommended.

5.4 IP Journal Database – IP Data Source

This chapter outlines the setup of a new data source for fault-tolerant Instant Payment (IP) architectures, specifically for IP-Journal databases. IP-Journal databases are used by the Central server module during instant payment message processing (if `InstPmtJrnPersistence` is used). They are (all) also used by the GUI when accessing the IP journals.

1. **Database Support**:

   The system accommodates up to four separate IP journal databases, each holding tables prefixed with `IPJRN_<ClientPrefix>` and `IPJRNADD_<ClientPrefix>`, along with a sequence called `IPMPS_SEQNO`.

2. **Database Types**:

   - **Local Database**: Stores configuration and address book data for server startup – and configuration operations. Address books may be required during message processing and should be cached then, see section `[MPS_HANDLER]` in the Central Server configuration
   - **Instant Payment Journal Database**: Up to four dedicated databases for IP journal tables.
   - **Optional Shared Database**: Contains shared Nexi/SIA FEMS data, used only if FEMS XS is the payment gateway.

3. **Migration Activities**:

   - Update the Central Server configuration with a new section `[IPJRN_DB_INTERFACE]` (see below) to specify the IP-Journal database;
   - Define the IP-Journal data sources in the GUI and apply them in the IP-Journal Views & Tasks, regardless of the number of data sources used;
   - No changes are needed for local database definitions;
   - Systems not using Nexi/SIA FEMS XS can remove the shared database configurations.
   - IP systems using Nexi/SIA FEMS XS do not need to change the shared database configuration

IP-Journal databases are used by the Central server module during instant payment message processing (if `InstPmtJrnPersistence` is used). They are (all) also used by the GUI when accessing the IP journals.

The shared database holds FEMS XS shared data in the following tables:

`SIAIPEP`	- Endpoint data
`SIAIPEPFEMS`	- Endpoint-FEMS data
`SIABXSESS`	- Business user session (BX) data
`SRVCFG, SRVCFG_EA`	- Endpoint and FEMS related configuration.

4. **Shared Database Usage**:

   It is recommended to have the shared database available at all times. However, if none of the activities described below occurs, the shared database may be taken offline during a brief period but should be taken online again as soon as possible. (`EP_SESSSION_KEY_LIFETIME` should be set to 0 to disable BOX-side session key renewal and FEMS IP administrative channel heartbeat interval may be set to 0 - disabled.)

   Please note, a shared database is also required when using FEMS XS in non-IP systems.

   Startup activities:

   - Start of BOX FEMS Instant Payment Administration channels, FEMS EP key change, FEMS BX session change (Log on, Subscribe, Open, and shut down).
   - Start of BOX Central server startup for FEMS EP and FEMS BX data cache
   - FEMS Endpoint and FEMS administration & configuration in GUI (Messaging Gateway Module administration & configuration on Enterprise Level)
   - SIA FEMS channel (IP and non-IP) display

- Change of session state, EP key change, BX session change.

Additional Information and Configuration Recommendations are available in the previous chapter

5.5 Instant Payment Journal Writer

The IP Journal Writer is configured in the Server and respective LCG. For all configuration details, please refer to the BOX Configuration Document. An example of the server configuration is given below.

IMPORTANT

If an IP Journal Writer, section `[IP_JOURNAL_WRITER]`, is configured, there must also be an `IPJRN_DB_INTERFACE` section.

1. Use the new section `[IPJRN_DB_INTERFACE]` in the Central Server configuration to specify the database where the IP-Journal tables reside for this specific server instance / location. Examples and configuration steps can be found in the following documentation: `box_InstantPayments_v3r26.pdf`
   In previous releases the shared database has been used for this purpose.
2. Ensure that IP-Journal data sources are defined within the GUI and apply them in IP-Journal Views & Tasks, regardless of the number of data sources used. In previous releases the shared database has been used for single data source IP Journals. So, an IP Journal data source needs to be defined, and the IP Journal View & Task definition needs to be renewed.

   Please note the following:

   - **Local database definitions need not to be changed.**
   - **IP systems not using Nexi/SIA FEMS XS may remove the shared database definitions. IP systems using Nexi/SIA FEMS XS do not need to change the shared database configuration.**

IP-Journal databases are used by the Central server module during instant payment message processing, if `InstPmtJrnPersistence` is used. They are (all) also used by the GUI when accessing the IP journals.

Server configuration example:

```
[IP_JOURNAL_WRITER]
  DATA_COMPRESSION_METHOD           NONE
  EXPIRATION_PERIOD                  4

[IPJRN_DB_INTERFACE]
 CONNECTION_TIMEOUT                  10
 CONNECTION_TIMEOUT_CHECK_INTERVAL   60
 DATABASE_NAME                       $$R$DB_DATABASE_IPJRN_NAME$
 DATABASE_PASSWORD                   $$R$DB_DATABASE_ENC_PWD$
 DATABASE_QUALIFIER                  $$R$DB_DATABASE_IPJRN_QUALIFIER$
 DATABASE_USERNAME                   $$R$DB_DATABASE_IPJRN_USER$
 DATA_SOURCE_NAME                    $$R$DB_DATABASE_IPJRN_NAME$
 DB_INTERFACE_LIBRARY                $$R$DB_DATABASE_LIBRARY$
 EXECUTION_TRACE                     3
 MAX_CONNECTIONS                     300
```

5.6 Shared Data Source

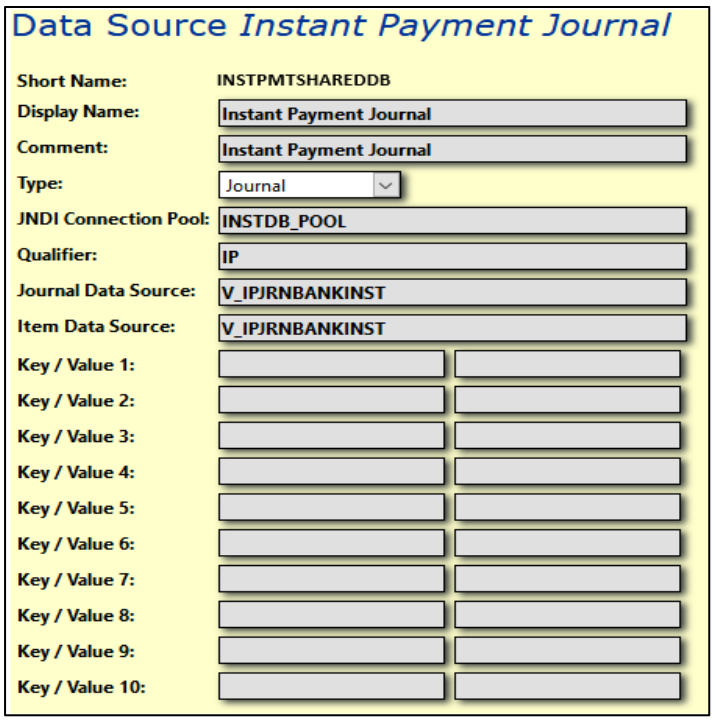
The already mentioned Instant Payment Journal is based on a view of shared data source tables.

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.

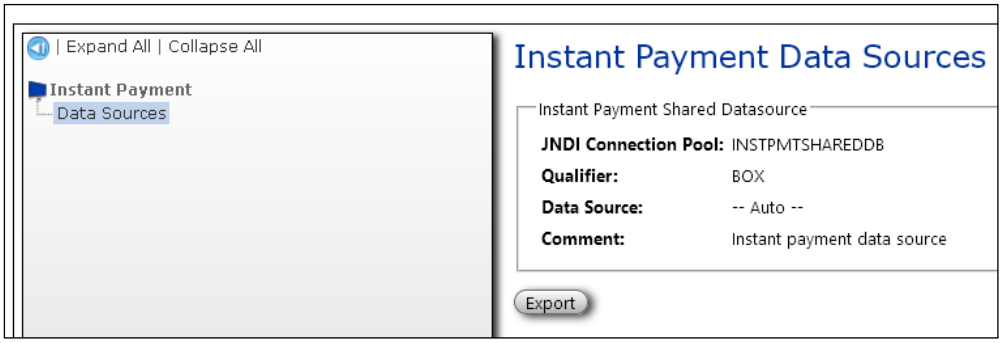
For the installation of the external database please follow the database installation procedure described in document box_inst_v3r23.pdf.

5.6.1 Creating a JNDI Connection

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



Picture 24 Creating a JNDI Connection



Picture 25 Configured Data Source

5.7 SwiftNet

5.7.1 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

  [IPJRN_DB_INTERFACE]
    CONNECTION_TIMEOUT = 10
    CONNECTION_TIMEOUT_CHECK_INTERVAL = 60
```

```
    DATABASE_NAME = $$R$DB_DATABASE_IPJRN_NAME$
    DATABASE_PASSWORD = $$R$DB_DATABASE_ENC_PWD$
    DATABASE_QUALIFIER = $$R$DB_DATABASE_IPJRN_QUALIFIER$
    DATABASE_USERNAME = $$R$DB_DATABASE_IPJRN_USER$
    DATA_SOURCE_NAME = $$R$DB_DATABASE_IPJRN_NAME$
    DB_INTERFACE_LIBRARY = $$R$DB_DATABASE_LIBRARY$
    EXECUTION_TRACE = 3
    MAX_CONNECTIONS = 300
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-MSG-SWIFT
    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
```

5.7.2 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 paid 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                    BOX-SIAIPRT-ADMIN
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
```

5.7.3 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.

5.7.3.1 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.

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 hex ascii 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).

Parameter	Default	Description
		If both LAU keys are set the following rule applies for HMAC calculation/verification: - 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 5.7-VI AGI Plugin Configuration Parameters

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

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-MSG-SWIFT
  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
```

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 (chapter0). The format of Output messages to be routed to a backend application is either MQMD + file data or MQMD + RFH2 Header + file data.

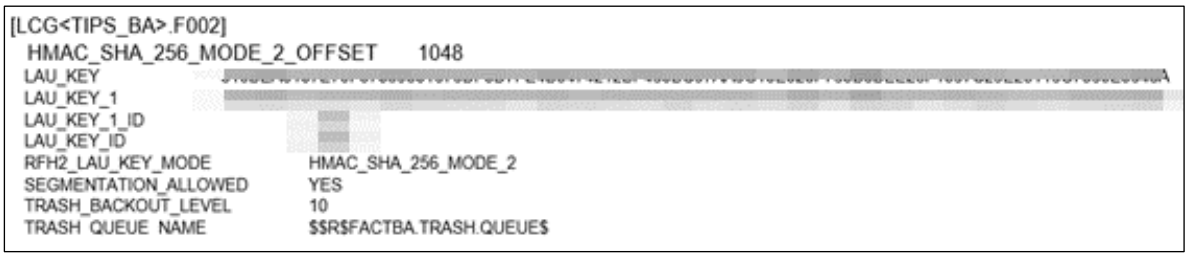
The format to be used depends on the backend application, i.e., on the format that the application expects.
If the backend application expects only MQMD and file data, the Backend Application LCG processes the BOX format message and hands it over to the backend application in the expected format.
If the backend application receives 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.

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



Picture 26 LCG TIPS F002 configuration excerpt

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`

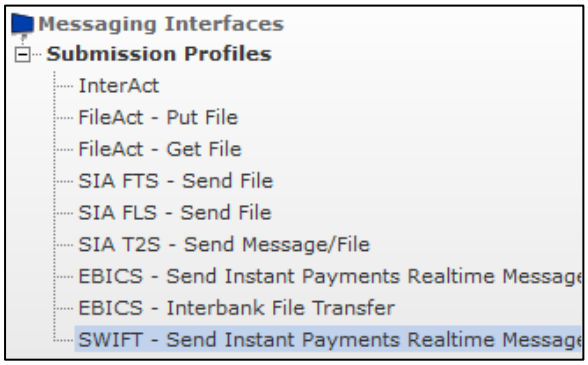
Table 5.7-VII Parameter eximf002 Configuration

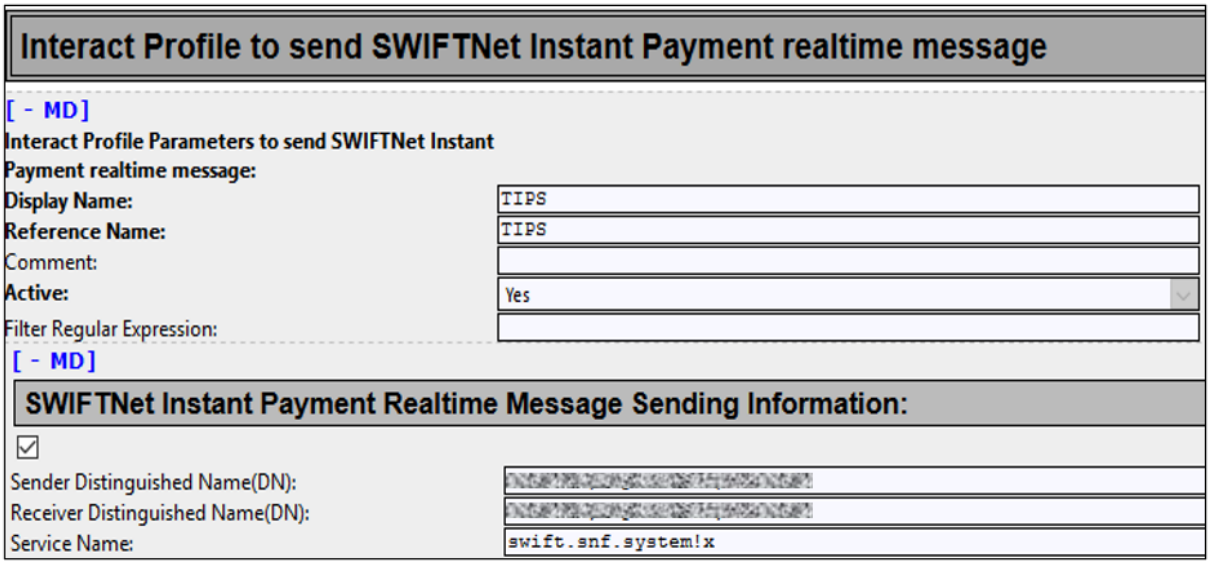
If using MQ-Cluster:

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

5.7.4 Message Enrichment Example

Submission Profile SWIFT





Picture 27 Submission Profiles – Example SWIFT Instant Payments

5.8 EBICS

5.8.1 Connectivity Channel to EBICS

Connectivity Channel Configuration Example

```
[LCG<TRAVIC_IP_IN_01>]
  CGTW_HOST                          ; <Prococol>:ModuleID,LCG-Number
  CHANNEL_TYPE                       EBICS-IB-FT-MI
  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)
```

5.8.2 VAN Gateway (EBICS/SWIFT) Configuration Options

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)

SWIFT

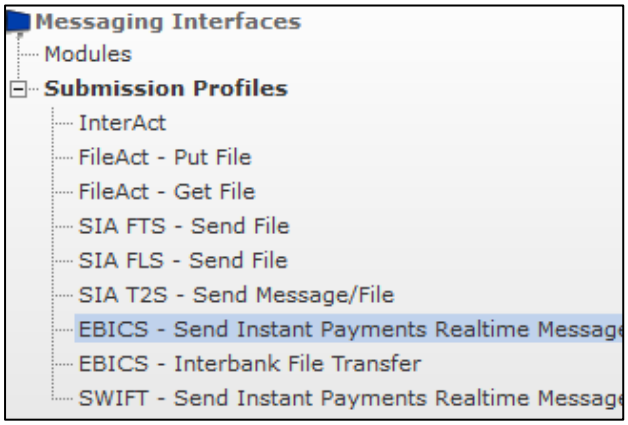
- Notification Request on/off: AGI configuration
- Technical Ack: requested Always
- Notification: depending on parameter `NOTIFICATION_REQUIRED`: on Error for separate AGI-Notification Request-Queue, Always for unset AGI-Notification Request-Queue
- `DELIVERY_REPORT_GENERATION`: Immediate

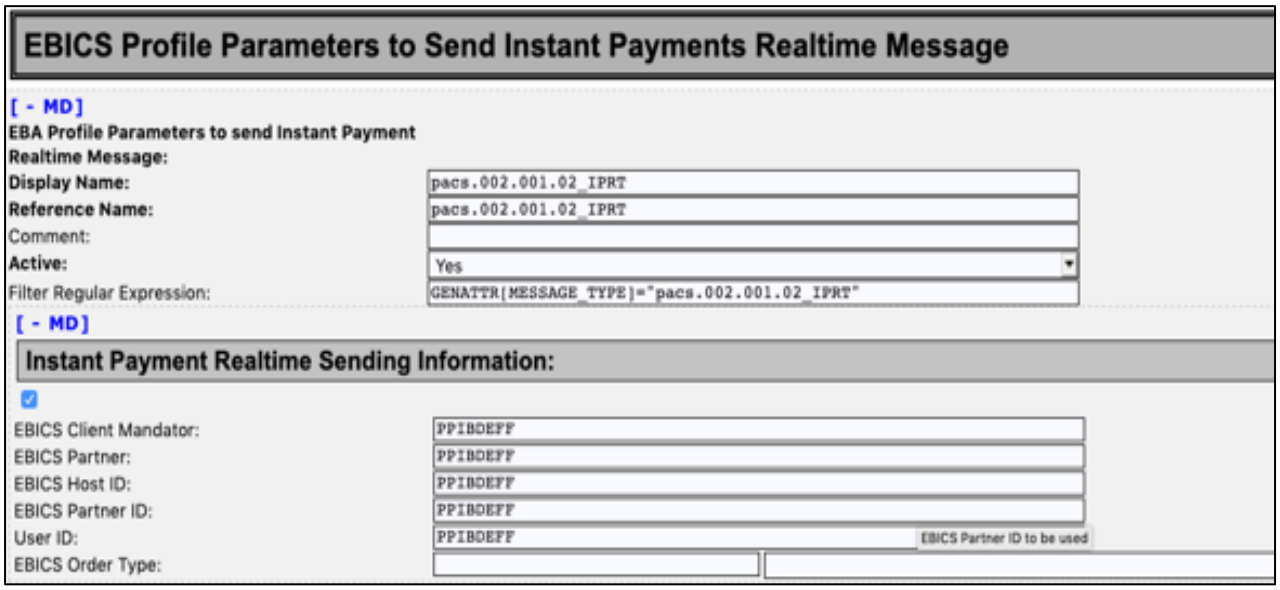
AGI Adapter processing for AGI Technical Responses:

- 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)

5.8.3 Message Enrichment Example

Submission Profile EBICS



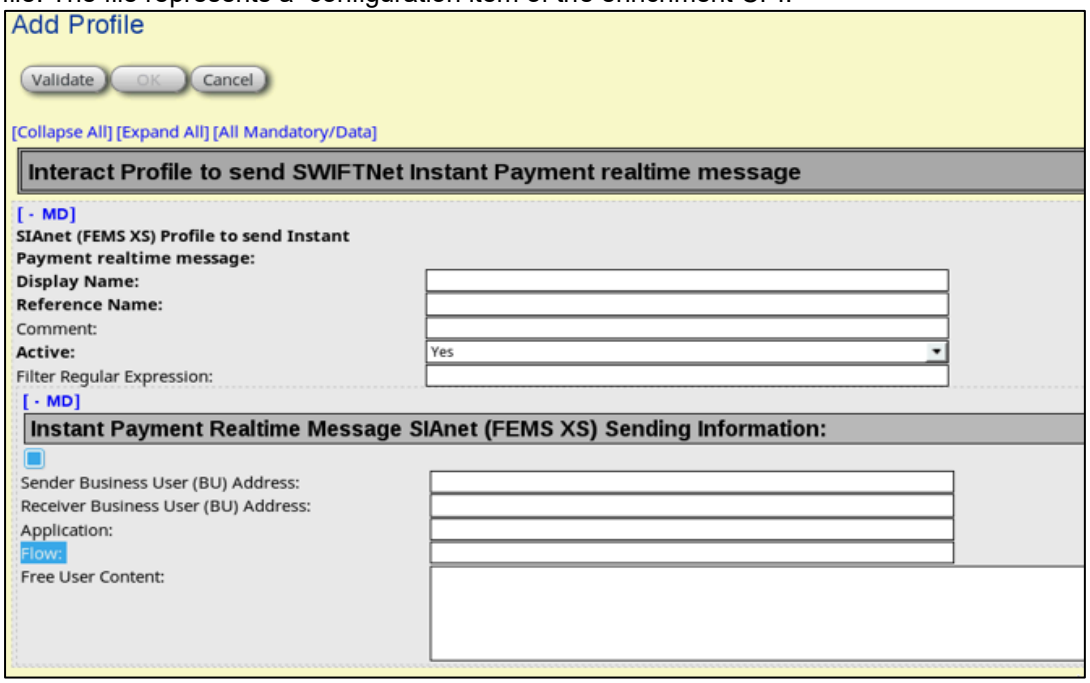


Picture 28 Submission Profiles – Example EBICS Instant Payments

5.9 SIANET

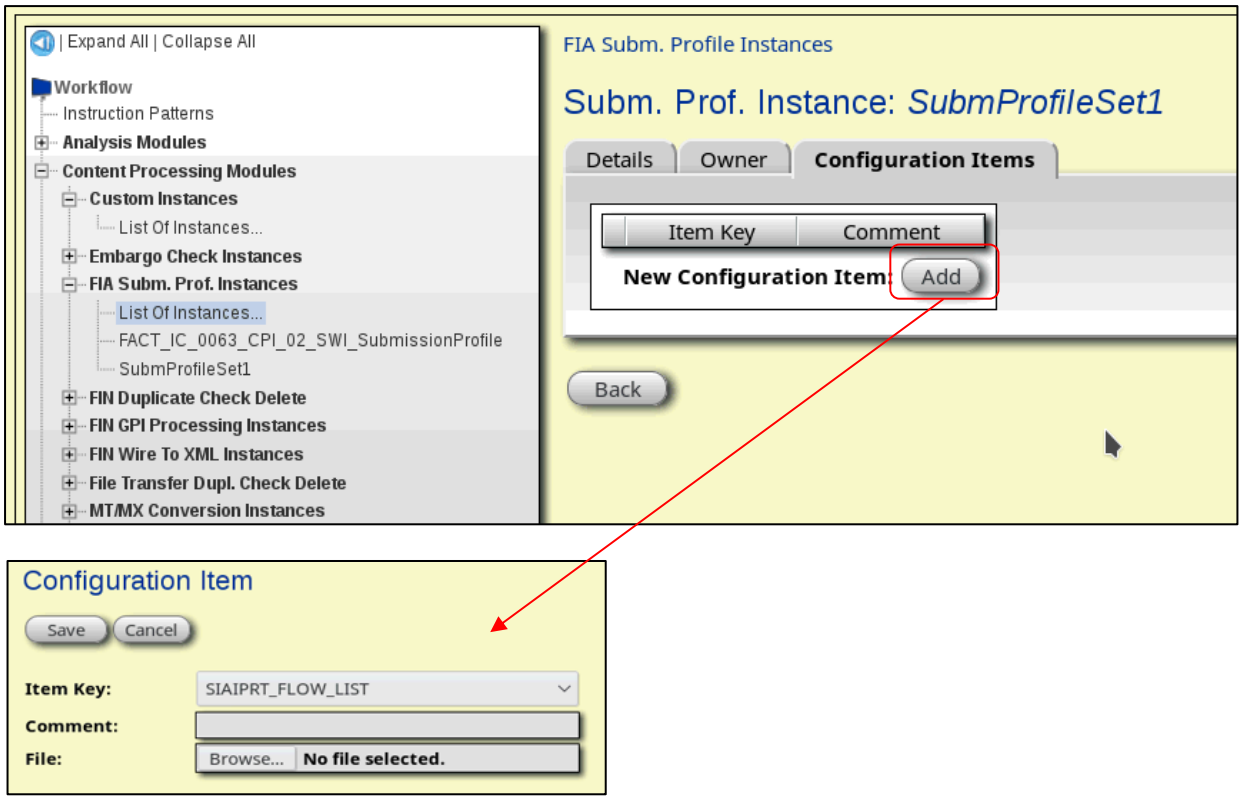
5.9.1 Configuration of "Submission Flow Enrichment" on SIANET

Every message type is related to a flow name. The enrichment Content Processing Instance (CPI) is adding this information to the transfer parameter. Either a fixed value is added, or a value read from a file. The file represents a configuration item of the enrichment CPI.



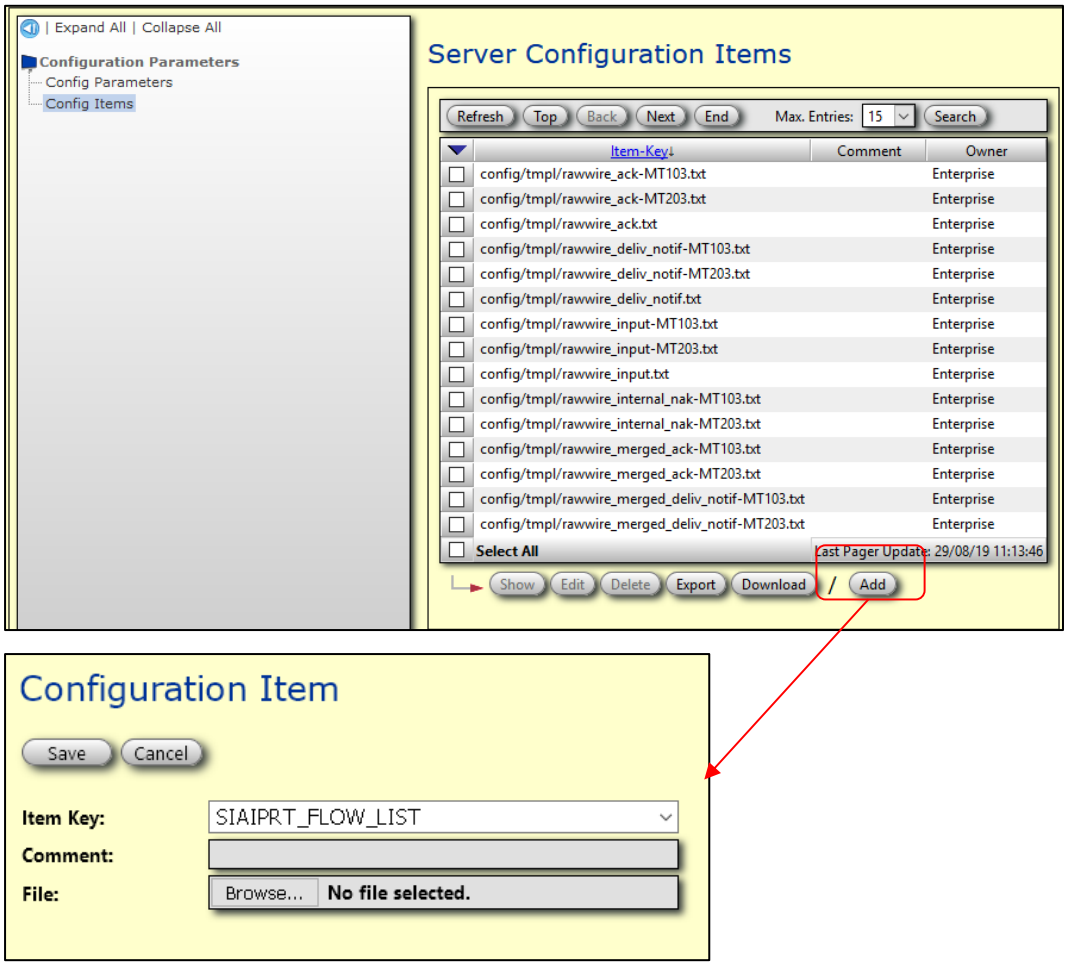
Picture 29 SIANET: Configuration of "Submission Flow Enrichment"

5.9.1.1 Adding the "SIA FEMS XS flow list"



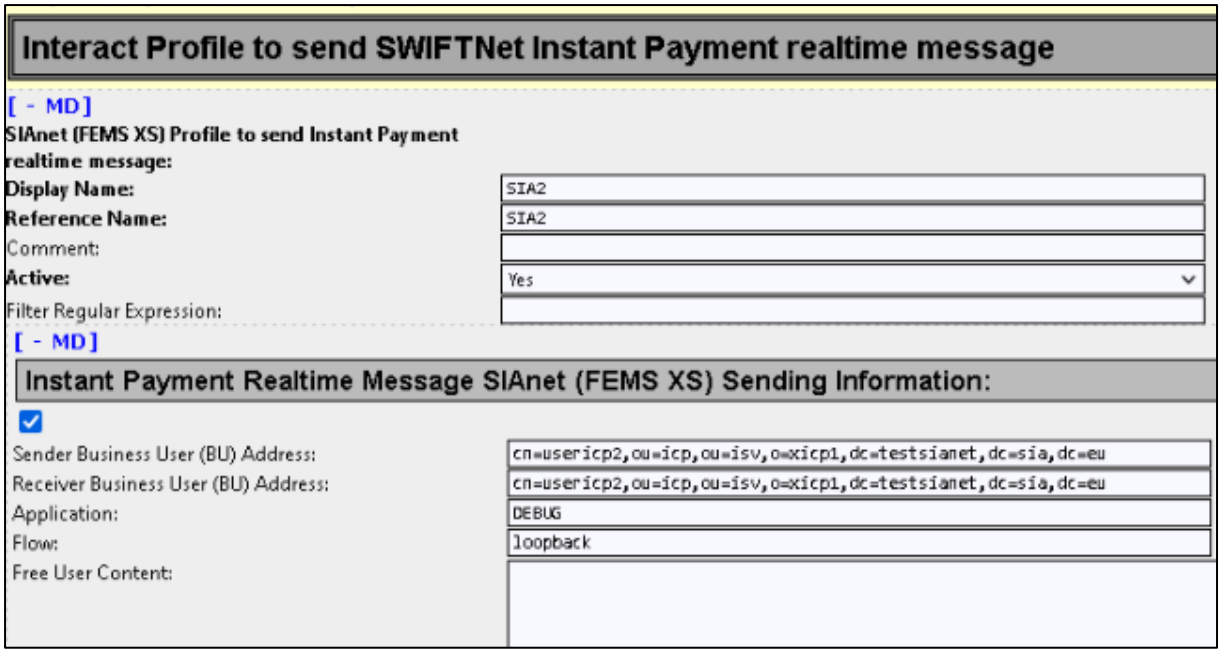
Picture 30 SIANET: Configuration of "Submission Flow Enrichment"

5.9.1.2 Adding Server Configuration Item (Example SIA Flow List)



Picture 31 Adding Special Flow Table for SIA

5.9.2 Message Enrichment Example



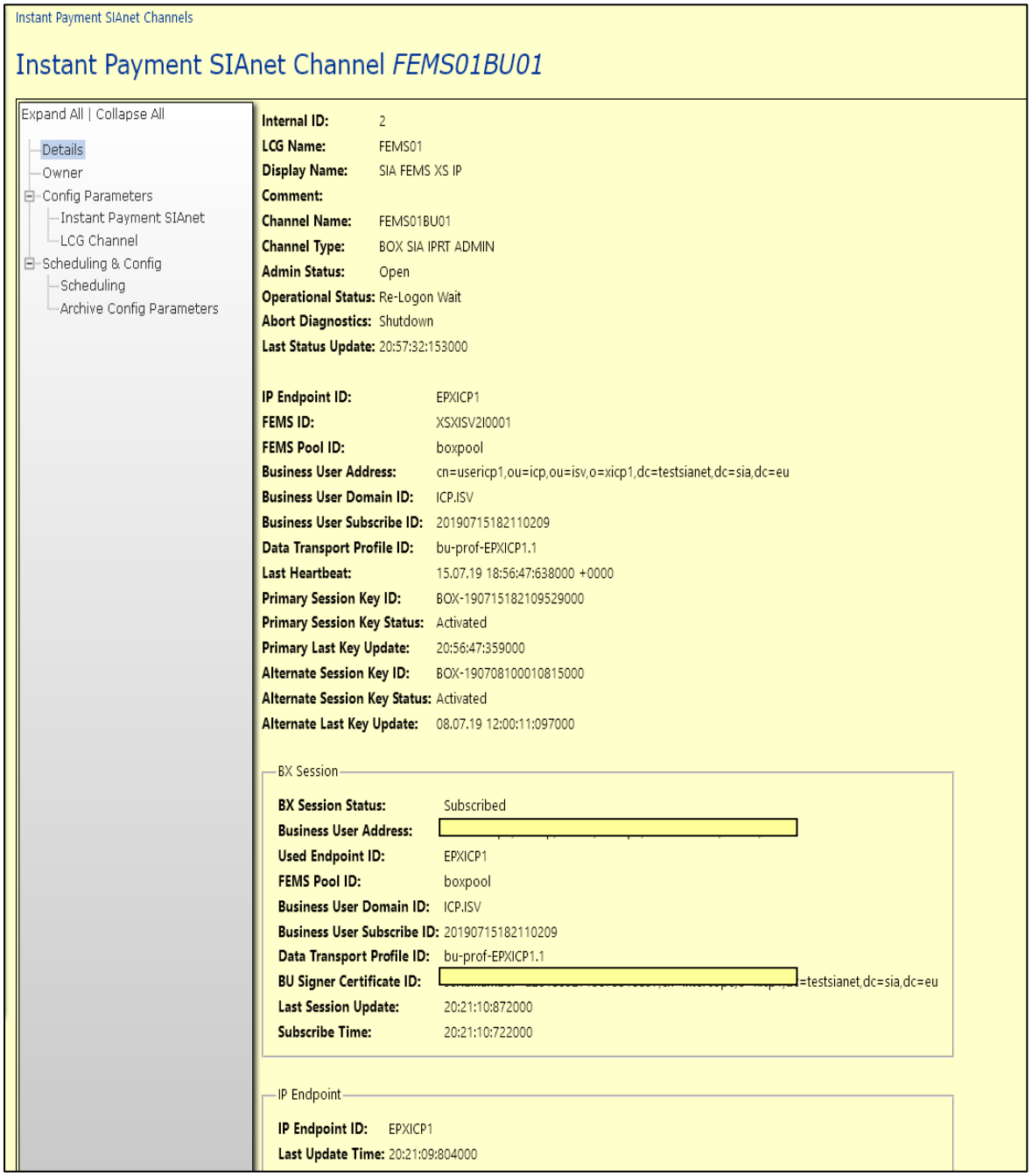
Picture 32 Submission Profiles – Example SIA Instant Payments

5.9.3 Messaging Interface to SIA FemsXS

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

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

Picture 33 Exemplary SIA Channel Setup 1



Picture 34 Exemplary SIA Channel Setup 2

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

Picture 35 Command Queue Set sharing (1 Set per FEMS)

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

```
[SIA_INSTPMT_CACHE]
  ENDPOINT_NAME                XXXXXX
```

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                      BOX-SIAIPRT-ADMIN
  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
```

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

5.10 SIC 5

Starting with Brelease 26 Fixpack 5 the SIC protocol version can be set in sections `[SIC-CHANNELNAME]` or `[SICIP-CHANNELNAME]` with parameter `SIC_PROTOCOL_VERSION`.

5.10.1 General

This chapter describes the implementation of SIC Instant Payments (`SIC IP; application protocol SIX Messaging Gateway V5`) in conjunction with SIC Real Time Gross Settlements (`RTGS; application protocol SIX Messaging Gateway V4`) in an exemplary setting. Both SIC payment systems are based on ISO20022 type messages.

To connect to SIC, BOX uses the path of a Messaging Gateway connection, via which most payments are made on the SIC platform. The API software for the connection is provided by BOX and the communication is realised within a closed IP-network, such as the SSFN.

To be able to send SIC messages, which are signed, the participant has to be authorised and must be eligible to send and receive. The authorization is checked based on a pair of keys for each transaction. A signature is calculated factoring in the key pair and is directly verified.

To comply with SIC 5 Instant Payments, the necessary BOX implementation is done for different parts, which may be configured to demand. Mostly adopting the already existing generic ISO20022 messaging structure, differences are implemented for Instant Payment messages. Supported message types are listed in the message catalogue to be viewed either by the BOX GUI or by file delivered with the respective package. The BOX message catalos also provides the version of the business protocol associated with the respective SIC/SIC-IP message version[1]:

1. `'SIC' domain module (SIC-RTGS Service and euroSIC-System): Protocol Version 4.9 and 4.10`

2. `'SICIP' (SIC-Instant Payments Service): Protocol Version 5.0 This chapter describes a set up option including in part SIC 4 RTGS and SIC 5 IP.`

On the following page, an overview of an exemplary SIC system highlights the main components. Messages sent via SIC are secured by signature and the communication by encryption. The SIX Advanced Security Server (SASS) provides the security.

5.10.2 BOX Specifics

General requirements - SIX and SASS components are only guaranteed for the

- use of supported Java version,

Java 11 used, minimum for BOX Messaging Gateway Modules

- operation systems Linux RedHat (x64) or Windows

- SIC/SIX provided software and dependencies as shown

`sic-supplement-17.0.7.zip:`

```
agrona-1.19.2.jar
annotations-3.0.1.jar
crypto-client2-5.1.1.jar
dagger-2.28.3.jar
javax.annotation-api-1.3.2.jar
javax.inject-1.jar
```

[1] For the latest version, please refer to the message catalogue.

```
jsr305-3.0.1.jar
log4j-api-2.19.0.jar
netty-buffer-4.1.97.Final.jar
netty-codec-4.1.97.Final.jar
netty-common-4.1.97.Final.jar
netty-handler-4.1.97.Final.jar
netty-resolver-4.1.97.Final.jar
netty-transport-4.1.97.Final.jar
netty-transport-native-unix-common-4.1.97.Final.jar
sic-adapter-core-17.0.7.jar
```

These files must be in the java class path, e.g., `SRV_JAVA_CLASSPATH` in the replacement file, usually `replace.rpl`.

In this specific set up Box is configured to transmit both SIC IP - and SIC RTGS messages to and from SIX, as well as to receive technical acknowledgments.

Receiving messages via MQ from a payment hub, BOX LCGs transfer these messages (`SIC-RTGS and SIC-IP`) to a dedicated workflow. Here, messages are processed, sent to specific address types, and delivered to `SIC-RTGS` or respectively to `SIC-IP`. For both the following address types are implemented:

- `Instant Payments Realtime SIC`
- `SIC`

All LCGs and the workflow are part of the BOX Server configuration. A sample configuration is given in chapter 5.10.3.[2]
The central instance to deal with all transmissions and receptions is the Messaging Interface (Gateway) linked to SIX via channels.

Channel-Concentrator

These channels may be combined in a "`Channel-Concentrator`" due to different requirements.

- CC0x - high availability – different instances (ha1, ha2) of SIC systems or customer interface ID's
- CC1x - system group – different SIC test systems
- RT9x - RTGS processing – using InstPmtJrnPersistence for RTGS messages

Please note the following Channel-Concentrator restrictions:

- No combination of Test or Production systems
- All concentrated channels must be authorized to process the IID of the message
- All concentrated channels must use the same BusinessServiceId
- All concentrated channels must use the same set of MQ-Objects
- Concentrated channels can be spread over different MIGTW, Nodes (BOX Installations) as long the used set of MQ-Objects is the same and available.
- Message routing inside the server to the Channel Concentrator can use any available field of the message.

Selecting a Channel Concentrator:

- The "SIC Communication ID" of a channel from Concentrator must match the "Receiver Communication Interface ID" of the message transfer parameters.
- The "Own Communication ID" of a channel from Concentrator must match the "Sender Communication Interface ID" of the message transfer parameters if not empty.

[2] For all parameters, please refer to the BOX configuration guide.

Selecting a channel inside the Channel Concentrator:

- The "SIC Communication ID" of the channel must match the "Receiver Communication Interface ID" of the message transfer parameters,
- The "Own Communication ID" is used/replaced by the configured ID of the used target channel
- An "Own Communication ID" can only be connected once:
- The customer has to organise the use of the same "Own Communication ID" on different nodes/BOX Installations or avoid configuring the same Own Communication ID on different nodes
- The server SICIP-MSG-LCG have to request "MQ Expiration Reports"

Channel initialization

To start, the channel must be initialized and the SIC Business Service must be set. If not done so, the dispatching of messages to the MI does not work correctly. A missing initialization is shown in the logged error below.

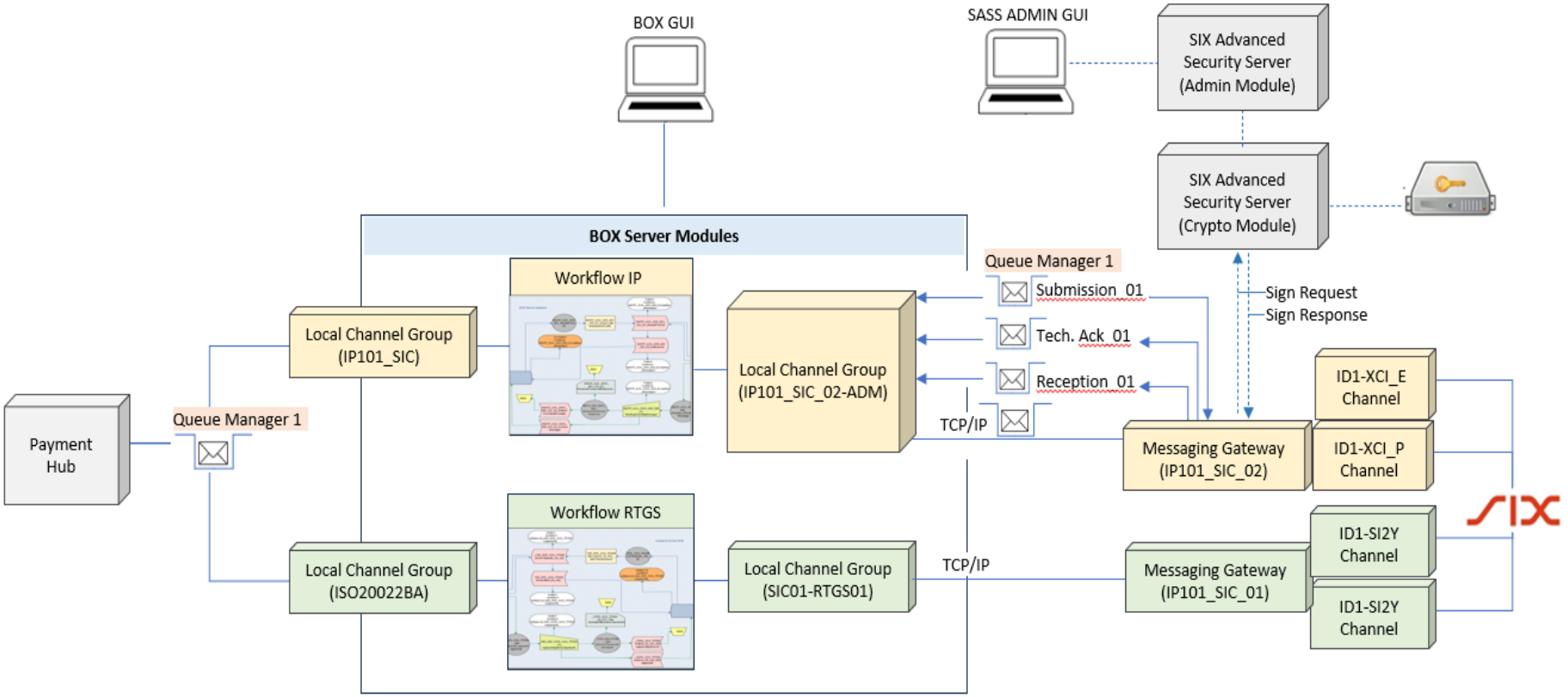
```
E: SICIP-CC01-MSG01: SIC5 IP LCG, Channel Concentrator AppGroup 'SICIP-
CC01-CBT', has empty SIC Business Service: Restart Server after all 'BOX-
SICIP-ADMIN'-channels have been initialized initially
```

To initialize, open the channel on a running system. There is no need to have a successful connection. Once done, please close the channel and restart the system.

SIC-IP/SIC-RTGS joined setup with the path message transactions take. Each transaction from a payment hub via BOX to SIX is authorized by a SASS Crypto Module before reaching the end point on the SIX platform. The System comprises a dedicated workflow with SIC-IP and SIC-RTGS Addresses as receiver at SIX.

`SIC-IP: address type Instant Payments Realtime SIC`

`SIC-RTGS: address type SIC`



Picture 36 Overview of SIC 5 Implementation

5.10.3 BOX Server LCG Configuration

The configuration of the BOX Server may be uploaded into the database or kept as ../config/mposerver.cfg file. Below excerpt reflects a possible SIC-RTGS and SIC-IP LCG configuration[3]:

```
[LCG<ISO50-IP101>]
  2PC_FOR_MESSAGES                          YES
  2PC_FOR_REPORTINGS                        YES
  APPLICATION_GROUP_NAME                    ISO50-IP101
  CGTW_HOST
  LCG_OWNER                                 $$R$PFX1$:$$R$PFX1$
  CHANNEL_TYPE                              IAFA-BA-INTERFACE
  DEFAULT_DELIVERY_COMPOSITION              0x0012101
  DISABLE_LCG                               NO
  SUPPORTED_ADDRESSTYPES                    IAFABA
  IMPORTER_COUNT                            1
  TRANSMITTER_COUNT                         1
  TRANSMITTER_BLOCKTHRESHOLD                0
  TRANSMITTER_UNBLOCKTHRESHOLD              0
  IP_JRN_WRITER                             ip_journal_writer

[LCG<ISO50-IP101>.PEXA]
  CREATOR_PREFIX                            $$R$PFX1$
  DEFAULT_CREATOR                           $$R$DEFAULT_CREATOR1$
  DEFAULT_IPS_SHORTLABEL                    I-01-iip
  DEFAULT_EXCEPTION_LABELPREFIX             $$R$PFX1$
  DEFAULT_EXCEPTION_SHORTLABEL              I-06-InternalFailure-Target-01-iip
  DEFAULT_MPS_INITMODE                      2
  SECURITY_FAILURE_LABELPREFIX              $$R$PFX1$
  SECURITY_FAILURE_SHORTLABEL               I-08-SecurityFailure-iip
  DELIVERY_MONITOR                          YES
  DEVICE_TYPE                               0xF002
  IMPORT_CHECK_CYCLE                        5
  MPS_PERSISTENCE_LEVEL                     InstPmtJrnPersistence
  MONITOR_CARRIER_DELIVERY                  NO
  PEXA_LIBRARY                              $$R$MQ_LIB_01$
  STORAGE_PERIOD                            12

[LCG<ISO50-IP101>.F002]
  DEFAULT_OUTBOUND_QUEUE                    IP101.DUMMY.71.DUMMY.OUT
  DEFAULT_OUTBOUND_QUEUE_MANAGER            $$R$QMGR_NAME_01$
  DEFAULT_REPLY_QUEUE                       IP101.BAE01.TO.ISO50.INB
  DEFAULT_REPLY_QUEUE_MANAGER               $$R$QMGR_NAME_01$
  DELIVERY_REPORT_GENERATION                0
  GENERATE_COMMAND_REPORT                   NO
  INBOUND_QUEUE                             IP101.BAE01.TO.ISO50.INB
  LOCAL_QUEUE_MANAGER                       $$R$QMGR_NAME_01$
  TRASH_BACKOUT_LEVEL                       1
  EXCEPTION_BACKOUT_LEVEL                   3
  TRASH_QUEUE_NAME                          IP101.ISO50.TO.IP101.TRS
  USE_DEAD_LETTER_QUEUE                     NO
  PLUGIN_LIBRARY_NAME                       expgi_iso20022ba

[LCG<ISO50-IP101>.F002.ISO20022BA_PLUGIN]
  EXTENDED_MESSAGE_TRACE                    $$R$EXTENDED_XXX_TRACE_01$
  DEFAULT_EXCEPTION_MESSAGE_TYPE            MX
  DEFAULT_EXCEPTION_FILE_TYPE               NONE
  INPUT_XSLT                                config/xslt/bae/ISO20022BA/BOX-IA-
Submission.xslt
  OUTPUT_XSLT                               config/xslt/bae/ISO20022BA/BOX-IA-
Reception.xslt
  PAYLOAD_MESSAGE_ROOT_NODE                 BackOfficeXchange
  VALIDATE_MESSAGE                          NO
```

[3] All parameters are listed and explained in the BOX Configuration Guide

```
  VALIDATION_FAILURE_PATTERN                    I-06-InternalFailure-Target-01-iip
  VALIDATION_FAILURE_PATTERN_PREFIX             $$R$PFX1$

[LCG<ISO50-IP101>.F002.ISO20022BA_PLUGIN.XML_PLUGIN]
  EXTENDED_PARSER_TRACE                         $$R$EXTENDED_XXX_TRACE_01$
  INCLUDE_NAMESPACE                             YES
  INCLUDE_SCHEMA                                YES
  INDENT_XML_NODES                              NO

[LCG<ISO50-IP101>.GTW_TRANSMISSION]
  WORK_DIRECTORY                                lcgwork
```

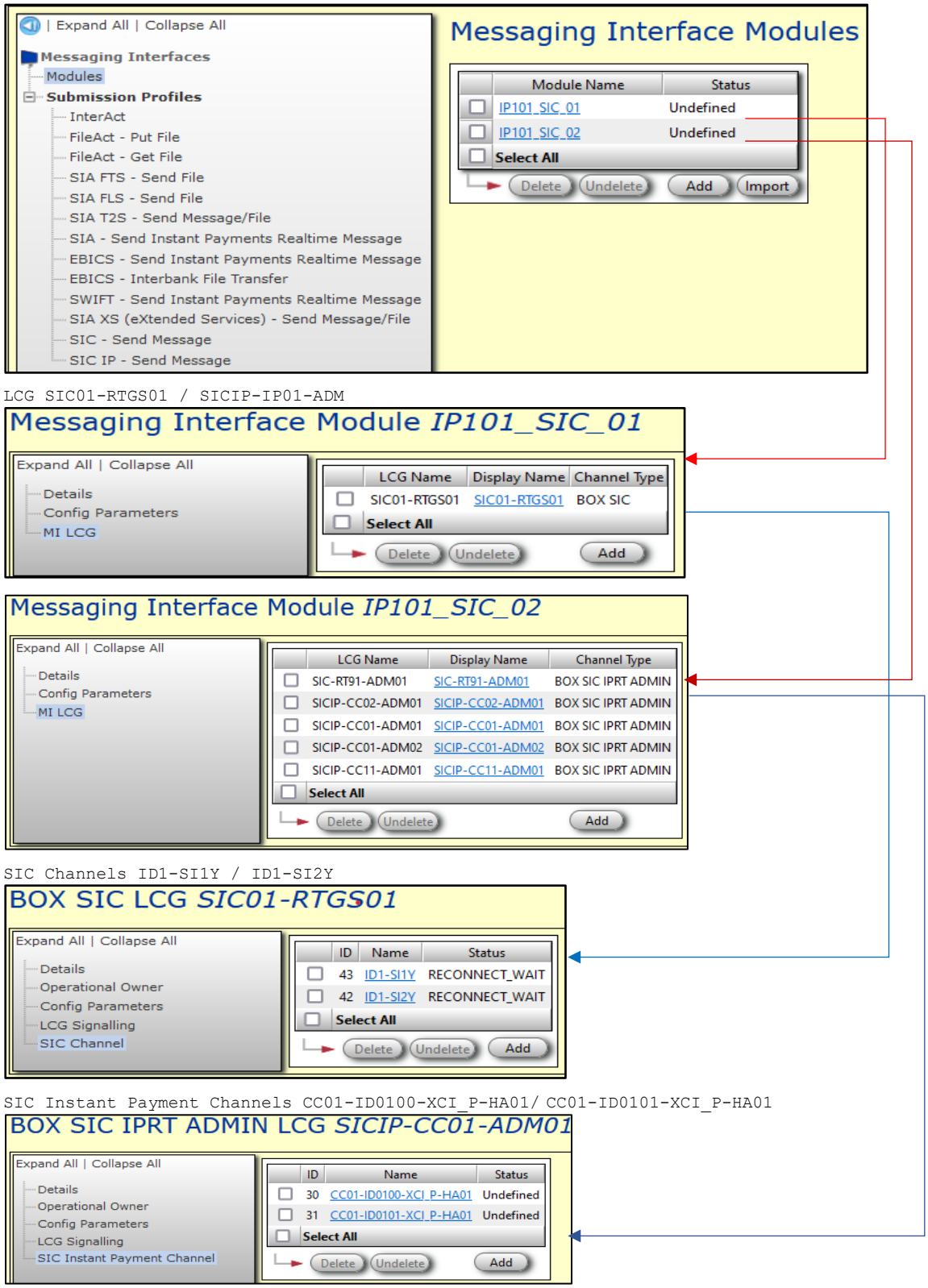
Below excerpt reflects a possible SIC-IP Admin and Message LCG configuration communication BOX Server <-> MI:

```
[LCG<SICIP-CC01-ADM01>]
  CHANNEL_TYPE                                  BOX-SICIP-ADMIN
  MGTW_HOST                                     T:2 ; <Prococol>:ModuleID

[LCG<SICIP-CC01-MSG01>]
  DISABLE_LCG                                   NO
  2PC_FOR_MESSAGES                              YES
  2PC_FOR_REPORTINGS                            YES
  APPLICATION_GROUP_NAME                        SICIP-CC01-CBT
  CACHE_CONTENT                                 YES
  CACHE_SIZE                                    50
  CARRIER_NAME                                  SICIP
  CHANNEL_TYPE                                  IP-RT-MSG-SIC
  IMPORTER_COUNT                                2
  IP_JRN_WRITER                                 ip_journal_writer
  LCG_OWNER                                     $$R$PFX1$:$$R$PFX1$
  MAX_ITEM_SIZE                                 1000
  MGTW_HOST                                     ;
  TRANSMITTER_BLOCKTHRESHOLD                    0
  TRANSMITTER_COUNT                             5
  TRANSMITTER_UNBLOCKTHRESHOLD                  0

[LCG<SICIP-CC01-MSG01>.PEXA]
  CREATOR_PREFIX                                $$R$PFX1$
  DEFAULT_CREATOR                               $$R$DEFAULT_CREATOR1$
  DEFAULT_EXCEPTION_SHORTLABEL                  O-06-InternalFailure-Target-01-iip
  DEFAULT_IPS_SHORTLABEL                        O-01-iip
  DEFAULT_MPS_INITMODE                          2
  DELIVERY_MONITOR                              YES
  DEVICE_TYPE                                   0xF002
  IMPORT_CHECK_CYCLE                            5
  MONITOR_CARRIER_DELIVERY                      NO
  MPS_PERSISTENCE_LEVEL                         InstPmtJrnPersistence
  PEXA_LIBRARY                                  $$R$MQ_LIB_01$ ; eximf002_cl
  STORAGE_PERIOD                                12

[LCG<SICIP-CC01-MSG01>.F002]
  ADDITIONAL_INBOUND_QUEUE_LIST                 IP101.MIX01.01.SRVXX.ACK
  DEFAULT_OUTBOUND_QUEUE                        IP101.SRVXX.01.MIX01.SUB
  DEFAULT_OUTBOUND_QUEUE_MANAGER                $$R$QMGR_NAME_01$
  DEFAULT_REPLY_QUEUE                           IP101.MIX01.01.SRVXX.ACK
  DEFAULT_REPLY_QUEUE_MANAGER                   $$R$QMGR_NAME_01$
  DELIVERY_REPORT_GENERATION                    DLV_REPORT_GEN_IMMEDIATE_IS_CARRIER
  EXCEPTION_BACKOUT_LEVEL                       5
  GENERATE_COMMAND_REPORT                       NO
  INBOUND_QUEUE                                 IP101.MIX01.01.SRVXX.REC
  LOCAL_QUEUE_MANAGER                           $$R$QMGR_NAME_01$
  PLUGIN_LIBRARY_NAME                           expgi_sic5ip
  EXPIRATION_TIMEOUT                            20 ; related to InstPmt timeout
  REQUEST_EXPIRATION_REPORT                     FullData
```

```
  RESPONSE_SERVER_AFFINITY                        Off
  TRASH_BACKOUT_LEVEL                             5
  TRASH_QUEUE_NAME                                IP101.SRVXX.01.MIX01.TRS

[LCG<SICIP-CC01-MSG01>.F002.SIC5IP_PLUGIN]
  ALWAYS_CREATE_MPS_FOR_POS_TECHACK               NO
  NEG_TECHACK_LABEL                               O-05-GRP-Switch-01-iip
  POS_TECHACK_LABEL                               ; using DEFAULT_IPS_SHORTLABEL


Below excerpt reflects a possible SIC RTGS LCG configuration  communication
BOX Server <-> MI:

[LCG<SIC01-RTGS01>]
  APPLICATION_GROUP_NAME                          SIC01-CBT-IP101
  CARRIER_NAME                                    SIX
  CHANNEL_TYPE                                    BOX-SIC
  IMPORTER_COUNT                                  3
  IMPORT_CHECK_CYCLE                              30
  MGTW_HOST                                       TCP:1,1
  TRANSMITTER_BLOCKTHRESHOLD                      0
  TRANSMITTER_COUNT                               1
  TRANSMITTER_UNBLOCKTHRESHOLD                    0

[LCG<SIC01-RTGS01>.PEXA]
  CREATOR_PREFIX                                  $$R$PFX1$
  DEFAULT_CREATOR                                 $$R$DEFAULT_CREATOR1$
  DEFAULT_IPS_SHORTLABEL                          OUT-01-iia
  DEFAULT_MPS_INITMODE                            2
  DELIVERY_MONITOR                                YES
  DEVICE_TYPE                                     BOX-SIC
  IMPORT_CHECK_CYCLE                              5
  MONITOR_CARRIER_DELIVERY                        NO
  PEXA_LIBRARY                                    eximf219_cl
  STORAGE_PERIOD                                  12

[LCG<SIC01-RTGS01>.F219]
  OUTPUT_PATTERN                                  OUT-01-iia
  OUTPUT_PATTERN_PREFIX                           $$R$PFX1$
```

5.10.4 Configuration Messaging Interfaces for SIC-RTGS and SIC-IP

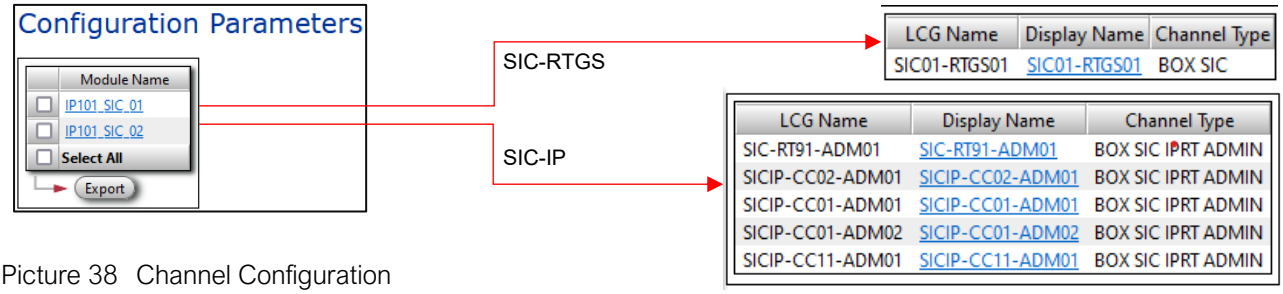
The configuration of the Messaging Interfaces (MIs) is done via GUI requiring the SYS Administrator. The set up entails the MI Module, the Local Channel Group with n number of channels, and a submission profile providing enrichment information for respective submissions. All done via GUI, the MI is implemented in 'Administration->Messaging Interfaces->Modules. A change set to enable editing is required for all steps:



Picture 37 Messaging Interface Implementation

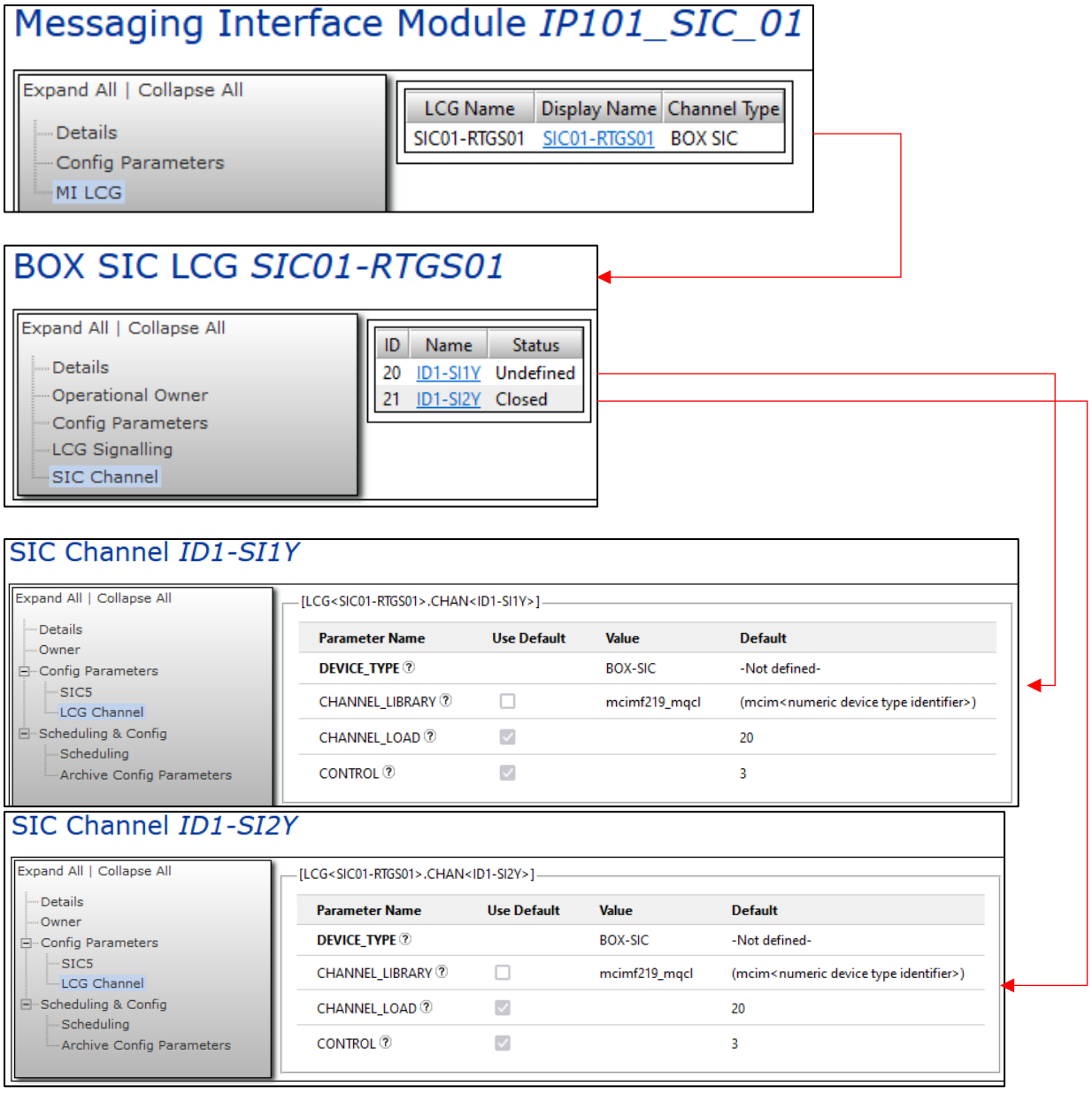
5.10.5 Channel Configuration for SIC-RTGS and SIC-IP

This chapter describes the configuration of the MI `IP101_SIC_01 (SIC-RTGS)` channels `ID1-SI1Y/ID1-SI2Y` with channel libraries `mcimf219_mqcl` for `SIC RTGS` and the MI `IP101_SIC_02` channels `CC01-ID0100-XCI_P-HA01/ CC01-ID0101-XCI_P-HA01` with channel library `mcimf218_mqcl` for `SIC Instant Payment:`



Picture 38 Channel Configuration

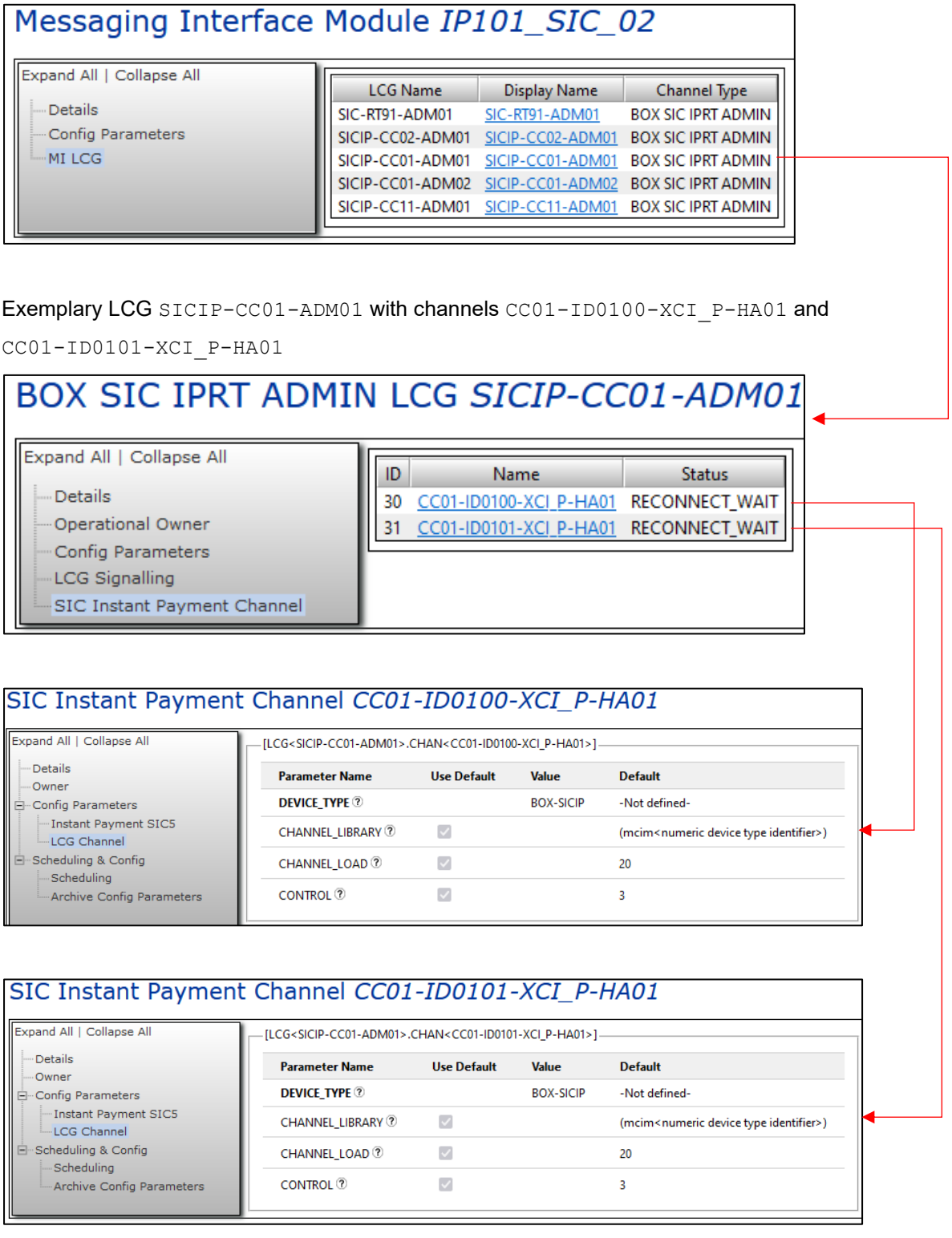
5.10.5.1 IP101.SIC.01



Picture 39 ID1-SI1Y and ID1SI2Y Channel Configuration

```
[ID1-SI1Y]
  BUSINESS_VERSION                          $$R$BUSINESS_VER_71$
  INITIAL_ADMIN_CHANNEL_STATUS              Closed
  LOOPBACK_MODE                             MODE1
  OWN_COMMINTF_ID                           $$R$OWN_COMMINTF_ID_71$
  SASS_BUSINESS_SERVICE_ID                  $$R$SASS_BUSI_SERVICE_ID_71$
  SASS_CLIENT_CERTSTORE                     config/sixx/certs/itct.p12
  SASS_CLIENT_CERTSTORE_STORE_PASSWORD      XXXXXXXXXXXXXXXXXXXXXXXXXXX
  SASS_HOST                                 $$R$SASS_HOST_01$
  SASS_PASSWORD                             $$R$SASS_PENC_01$
  SASS_PORT                                 10002
  SASS_PRODUCTION_FLAG                      NO
  SASS_USERNAME                             $$R$SASS_USER_01$
  SIC_ADAPTER_TRACE_MODE                    TEXT
  SIC_ADAPTER_TRACE_PATH                    logs/sic-rtgs
  SIC_COMMINTF_ID                           $$R$SIC_COMMINTF_ID_71_T01$
  SIC_HOST                                  $$R$SIC_HOST_71_T01$
  SIC_PORT                                  4002

[ID1-SI2Y]
  BUSINESS_VERSION                          $$R$BUSINESS_VER_71$
  LOOPBACK_MODE                             MODE1
  OWN_COMMINTF_ID                           $$R$OWN_COMMINTF_ID_71$
  SASS_BUSINESS_SERVICE_ID                  $$R$BUSINESS_VER_71$
  SASS_CLIENT_CERTSTORE                     config/sixx/certs/itct.p12
  SASS_CLIENT_CERTSTORE_STORE_PASSWORD      XXXXXXXXXXXXXXXXXXXXXXXXXXX
  SASS_HOST                                 $$R$SASS_HOST_01$
  SASS_PASSWORD                             $$R$SASS_PENC_01$
  SASS_PORT                                 10002
  SASS_PRODUCTION_FLAG                      NO
  SASS_USERNAME                             $$R$SASS_USER_01$
  SIC_ADAPTER_TRACE_MODE                    TEXT
  SIC_ADAPTER_TRACE_PATH                    logs/sic-rtgs
  SIC_COMMINTF_ID                           $$R$SIC_COMMINTF_ID_71_PRD$
  SIC_HOST                                  $$R$SIC_HOST_71_T01$
  SIC_PORT                                  4001

[LCG<SIC01-RTGS01>]
  CHANNEL_TYPE                              BOX-SIC

[LCG<SIC01-RTGS01>.CHAN<ID1-SI1Y>]
  CHANNEL_LIBRARY                           mcimf219_mqcl
  DEVICE_TYPE                               BOX-SIC

[LCG<SIC01-RTGS01>.CHAN<ID1-SI2Y>]
  CHANNEL_LIBRARY                           mcimf219_mqcl
  DEVICE_TYPE                               BOX-SIC

[MPO_MGTW]
  CHANNEL_CONFIG                            @IP101_SIC_01
```

5.10.5.2 IP101_SIC2



Exemplary LCG `SICIP-CC01-ADM01` with channels `CC01-ID0100-XCI_P-HA01` and `CC01-ID0101-XCI_P-HA01`

Picture 40 SIC-IP Channel Instant Payment SIC5 Configuration

Excerpt Configuration for LCG `SICIP-CC01-ADM01`

```
[CC01-ID0101-XCI_P-HA01]
  BUSINESS_VERSION                          $$R$SIP_P100_BUSI_VERSION$
  INITIAL_ADMIN_CHANNEL_STATUS              Closed
  LOCAL_QUEUE_MGRNAME                       $$R$QMGR_NAME_01$
  LOOPBACK_MODE                             MODE1
  OWN_COMMINTF_ID                           $$R$SIP_0101_OWN_COMIF_ID$
  RECEPTION_WRITE_QUEUE_NAME                IP101.MIX01.01.SRVXX.REC
  SASS_BUSINESS_SERVICE_ID                  $$R$SIP_S101_BUSI_SERVICE_ID$
  SASS_CLIENT_CERTSTORE                     config/sixx/certs/TCI_ITCT00.p12
  SASS_CLIENT_CERTSTORE_STORE_PASSWORD      1DB3F082D5EC764A82160B982B1DB025
  SASS_HOST                                 $$R$SIX_S101_HOST$
  SASS_PASSWORD                             $$R$SIX_S101_PENC$
  SASS_PORT                                 $$R$SIX_S101_PORT$
  SASS_PRODUCTION_FLAG                      $$R$SIX_S101_PRD_FLAG$
  SASS_USERNAME                             $$R$SIX_S101_USER$
  SIC_ADAPTER_TRACE_MODE                    TEXT
  SIC_ADAPTER_TRACE_PATH                    logs/sic-ip
  SIC_COMMINTF_ID                           $$R$SIP_P100_SIC_COMIF_ID$
  SIC_HOST                                  $$R$SIP_P1H1_HOST$
  SIC_PORT                                  $$R$SIP_M1H1_PORT$
  SUBMISSION_READ_QUEUE_NAME                IP101.SRVXX.01.MIX01.SUB
  TECHNICAL_ACK_WRITE_QUEUE_NAME            IP101.MIX01.01.SRVXX.ACK


[CC01-ID0101-XCI_P-HA02]
  BUSINESS_VERSION                          CURRENT
  INITIAL_ADMIN_CHANNEL_STATUS              Closed
  LOCAL_QUEUE_MGRNAME                       $$R$QMGR_NAME_01$
  LOOPBACK_MODE                             MODE1
  OWN_COMMINTF_ID                           $$R$SIP_0101_OWN_COMIF_ID$
  RECEPTION_WRITE_QUEUE_NAME                IP101.MIX01.01.SRVXX.REC
  SASS_BUSINESS_SERVICE_ID                  $$R$SIP_S101_BUSI_SERVICE_ID$
  SASS_CLIENT_CERTSTORE                     config/sixx/certs/TCI_ITCT00.p12
  SASS_CLIENT_CERTSTORE_STORE_PASSWORD      1DB3F082D5EC764A82160B982B1DB025
  SASS_HOST                                 $$R$SIX_S101_HOST$
  SASS_PASSWORD                             $$R$SIX_S101_PENC$
  SASS_PORT                                 10002
  SASS_PRODUCTION_FLAG                      NO
  SASS_USERNAME                             $$R$SIX_S101_USER$
  SIC_ADAPTER_TRACE_MODE                    TEXT
  SIC_ADAPTER_TRACE_PATH                    logs/sic-ip
  SIC_COMMINTF_ID                           $$R$SIP_P100_SIC_COMIF_ID$
  SIC_HOST                                  $$R$SIP_P1H2_HOST$
  SIC_PORT                                  $$R$SIP_M3H1_PORT$
  SUBMISSION_READ_QUEUE_NAME                IP101.SRVXX.01.MIX01.SUB
  TECHNICAL_ACK_WRITE_QUEUE_NAME            IP101.MIX01.01.SRVXX.ACK


[LCG<SICIP-CC01-ADM01>]
  CHANNEL_TYPE    BOX-SICIP-ADMIN


[LCG<SICIP-CC01-ADM01>.CHAN<CC01-ID0100-XCI_P-HA01>]
  DEVICE_TYPE    BOX-SICIP


[LCG<SICIP-CC01-ADM01>.CHAN<CC01-ID0101-XCI_P-HA01>]
  DEVICE_TYPE    BOX-SICIP


[MPO_MGTW]
  CHANNEL_CONFIG    $DB:IP101_SIC_02
```

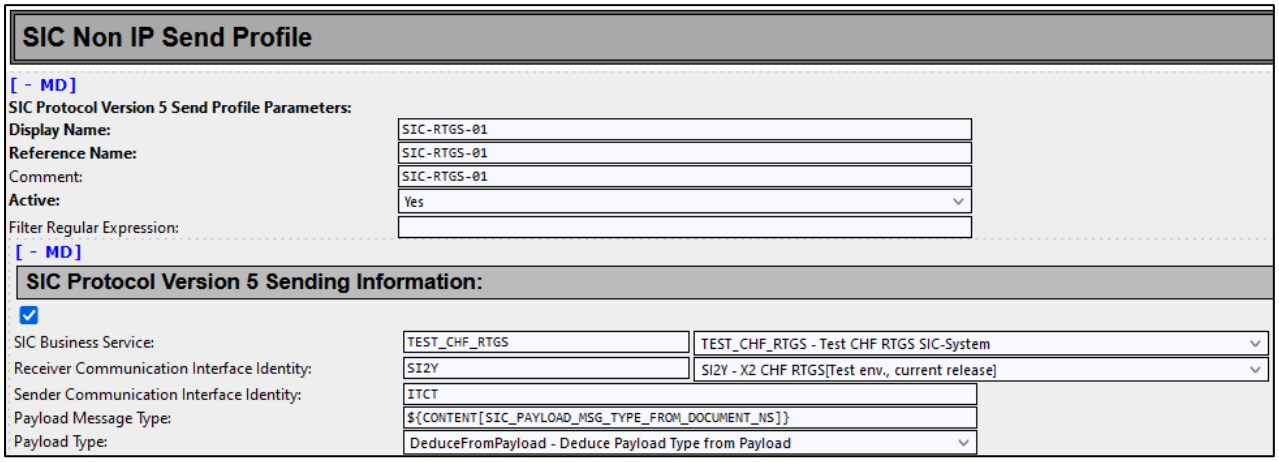
5.10.6 Message Enrichment Example

The enrichment of SIC-RTGS - and SIC-IP messages is done via submission profiles configured in the Messaging Interface set up (marked in red):

Picture 41 Submission Profiles for SIC-RTGS and SIC-IP

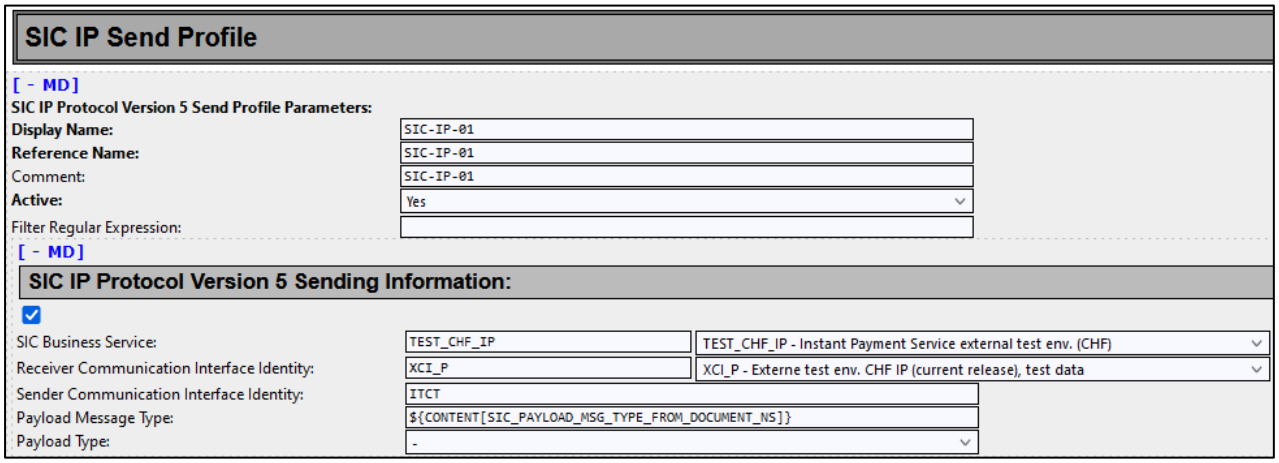
Exemplary Submission Profiles ‘SIC – Send Message’ and ‘SIC IP – Send Message’:

SIC-RTGS



Picture 42 Exemplary Submission Profile SIC-RTGS

SIC-IP



Picture 43 Exemplary Submission Profile SIC-IP

5.10.7 General Configuration Options for SIC 5

For all configuration options applies, that the latest developments and further information are to be found in the respective documentation:

- Box Workflow
- BOX Plugins
- BOX Admin Guide

1. **Submission profile enrichment CPI**
   The `cpi_fiasubprof content processing module` enriches the transfer parameters of SIC 5 (IP) messages.

   New replacement tokens are available for SIC 5 (IP):

Replacement Token	Description
`GENATTR[SIC_BUSINESS_SERVICE_ID]`	to get the SIC business service id
`GENATTR[SIC_RECEIVER_COMM_INTF_ID]`	to get the receiver comm interface id
`GENATTR[SIC_SENDER_COMM_INTF_ID]`	to get the sender comm interface id
`GENATTR[SIC_XML_MESSAGE_TYPE]`	to get the XML payload type
`CONTENT[SIC_PAYLOAD_MSG_TYPE_FROM _DOCUMENT_NS]`	to set the payload message type from the document namespace

2. **GRTA**
   It is possible to process SIC 5 messages via the GRTA with the following type sets:

Type Set	Description
`SIC 5`	type set to MX in GRTA
`SIC 5 Payment`	type set to Payment in GRTA
`SIC % IP`	type set to MX (IPRT) in GRTA

3. **Analyse 1**
   All available aliases are listed in the respective documentation, box_analysis1_<version>.pdf

4. **ISO20022BA Plugin**

   SIC 5
   To send SIC 5 messages, the xslt must create a `SICSendTransferParameters node` or a `SICMsgAttributeSet node` (see BOXMessage.xsd, mpo_pexai.xsd). The module SIC is determined by the `Document node namespace (`contains `.ch.` or `.chsepa.)`. If the Document namespace is not correct, the module SIC can be forced by setting the `module attribute` of the IGXM node to `SIC`.

   SIC 5 Instant Payment

   To send SIC 5 IP messages the xslt must create a `SICIPSendTransferParameters node` or an `MXSICIPRTAttributeSet node` (see BOXMessage.xsd, mpo_paxai.xsd).

   The module SICIP will be determined by the Document node namespace (contains .ch.)

   If the Document namespace is not correct the module `SICIP` can be forced by setting the module attribute of the IGXM node to `SICIP`.

5. **BOX Archiver**
   Add `MXSIC, PMTSIC` to the list of application attribute types (config parameter `SUPPORTED_MESSAGE_TYPES`) if SIC 5 (Payment) messages are to be archived.

```
[BOX_ARCHIVER]
SUPPORTED_MESSAGE_TYPES          MXSIC, PMTSIC, ..
```

6. **Generic XML Archiver**
   Set the config parameter `APPLICATION_ATTRIBUTE_TYPE` to `MXSIC` to archive SIC 5 Messages or to `PMTSIC` to archive SIC 5 payment messages.

```
[GENFILEEXP_ARCHIVER]
    GENFILEEXP_ARCHIVER             MXSIC
```

7. **MP BOX Archive XML Exporter**
   The tool also exports SIC 5 payment messages from the archive. The BAXMLExportRoot.xsd schema file is updated.

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 list of these restricted counter parties in the document "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

6.1.1 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.

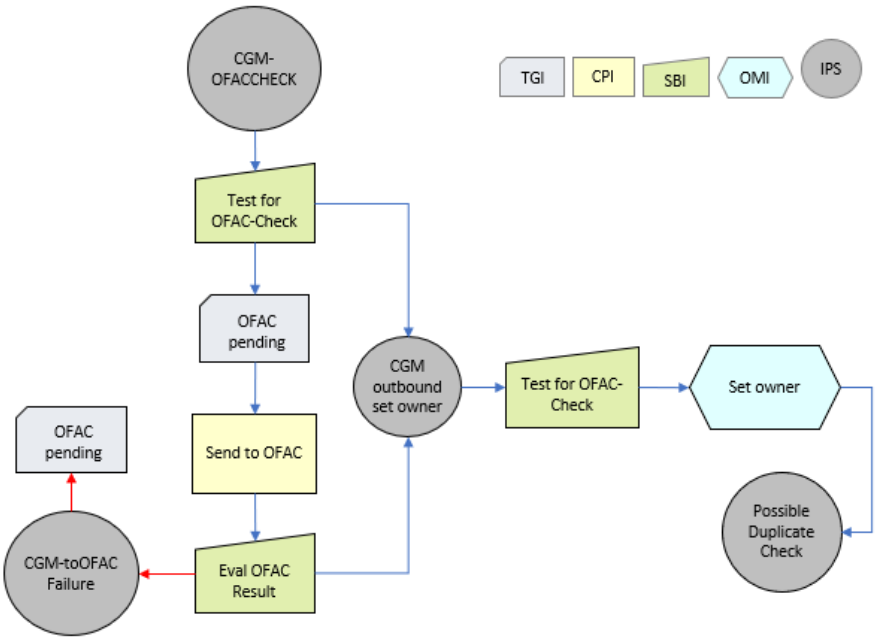
6.2 Workflow Concept

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



Picture 44 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 “`RouteToFIN_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.

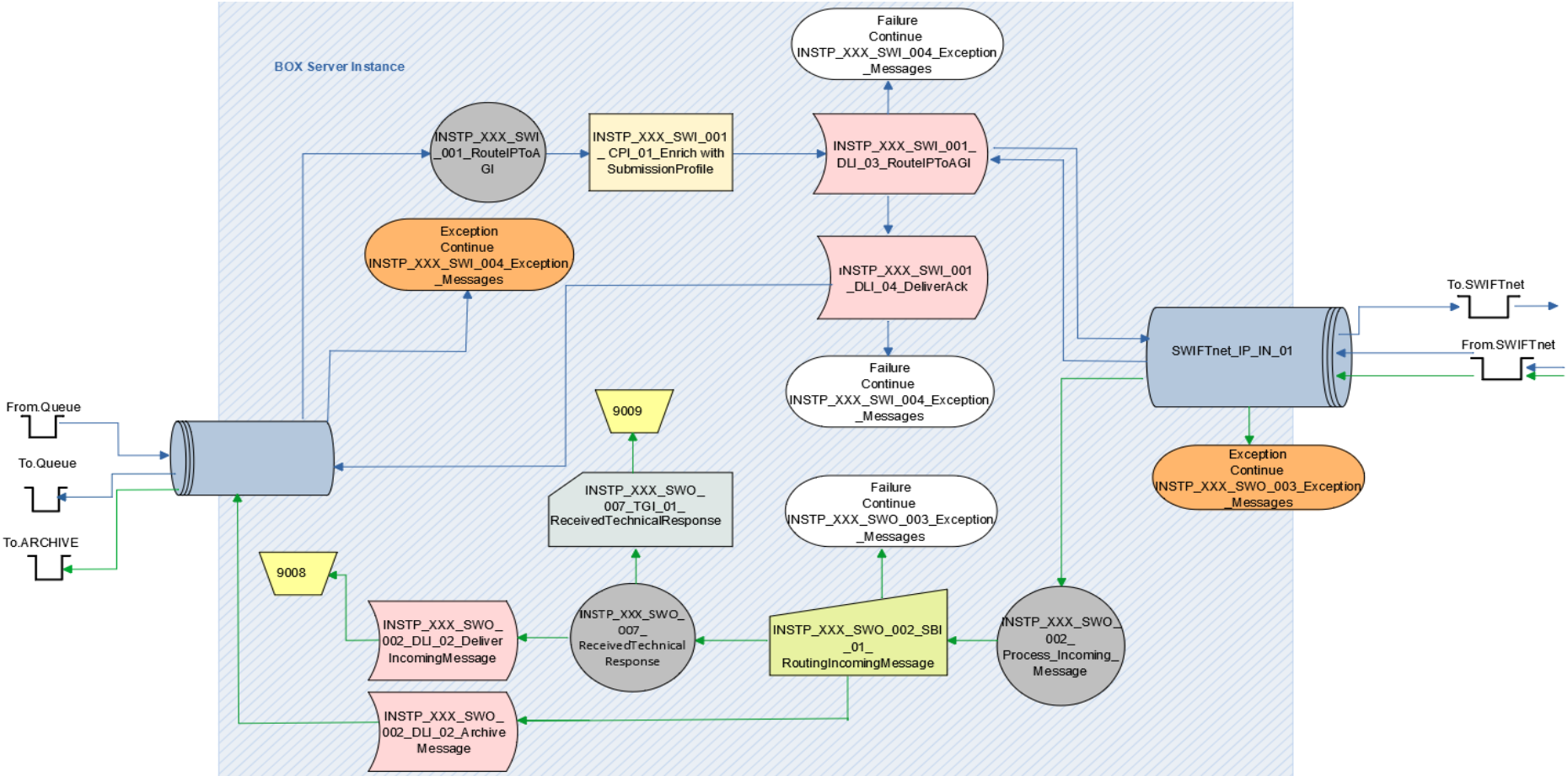
For manual testing, a Pacs.008.001.02 message is written to a backend queue, which is read by BOX.

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>
```

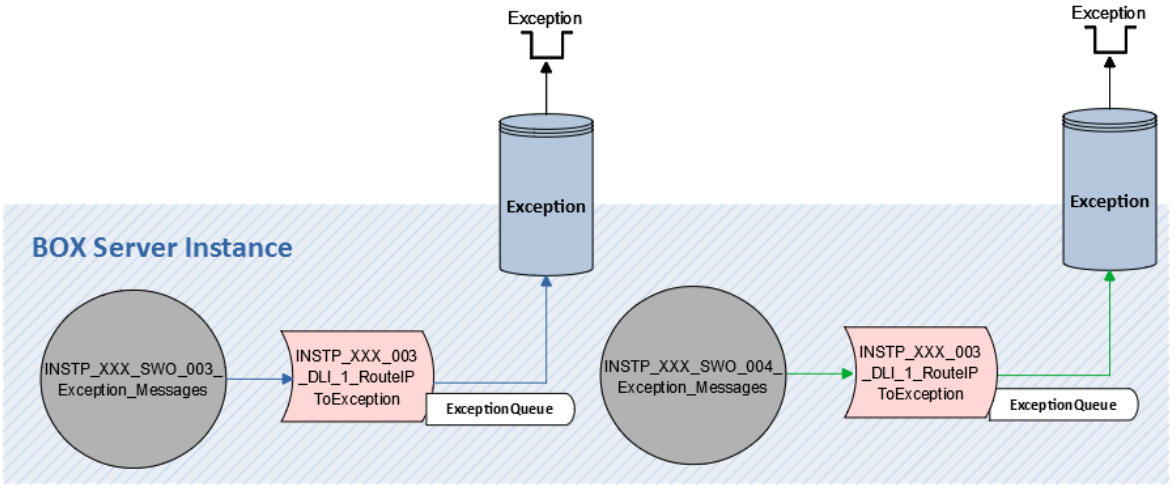
8 Workflow

8.1 Exemplary SWIFTnet Workflow of an Archive-Persistence Mode



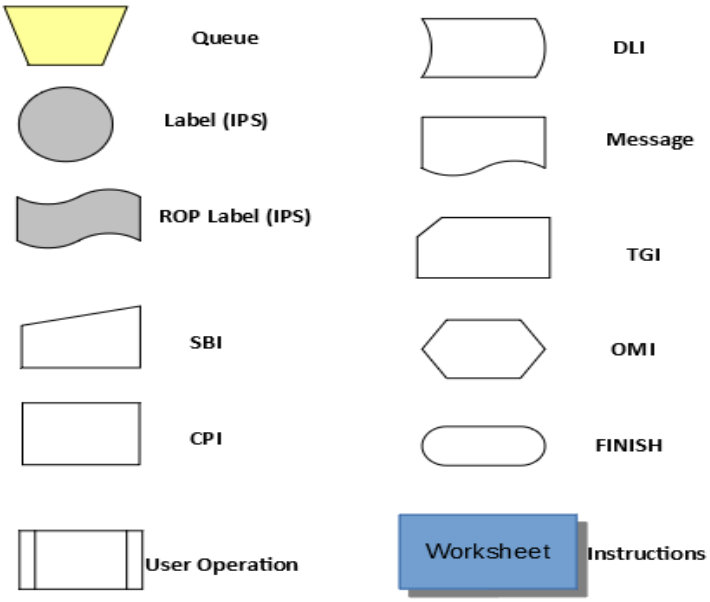
Picture 45 Overview SWIFTnet Instant Payment Workflow (TIPS)

8.2 Message Exception Workflow



Picture 46 Overview Exception Message Workflow

8.3 Signs and Symbols



Picture 47 Workflow Signs and Symbols

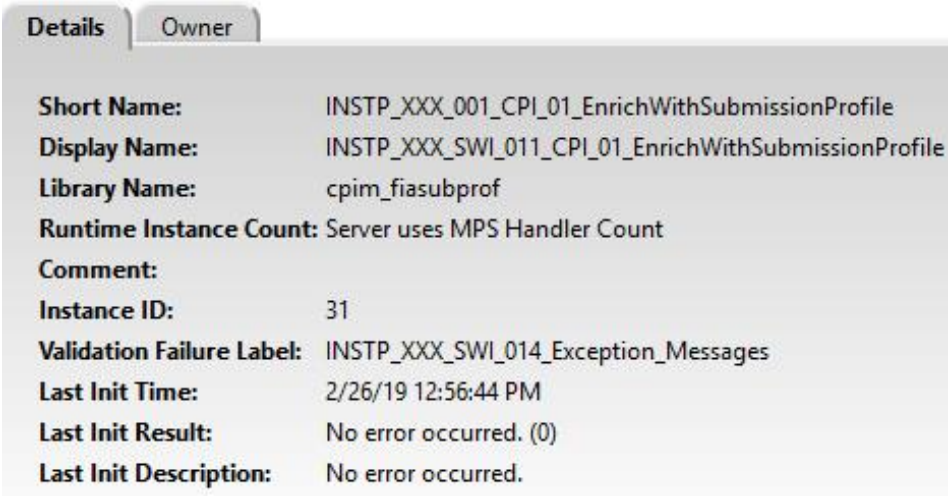
8.4 Instruction Patterns

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
```



Picture 48 Content Processing Modules-FIA Subm. Prof. Instance: Enrichment with Submission Profile

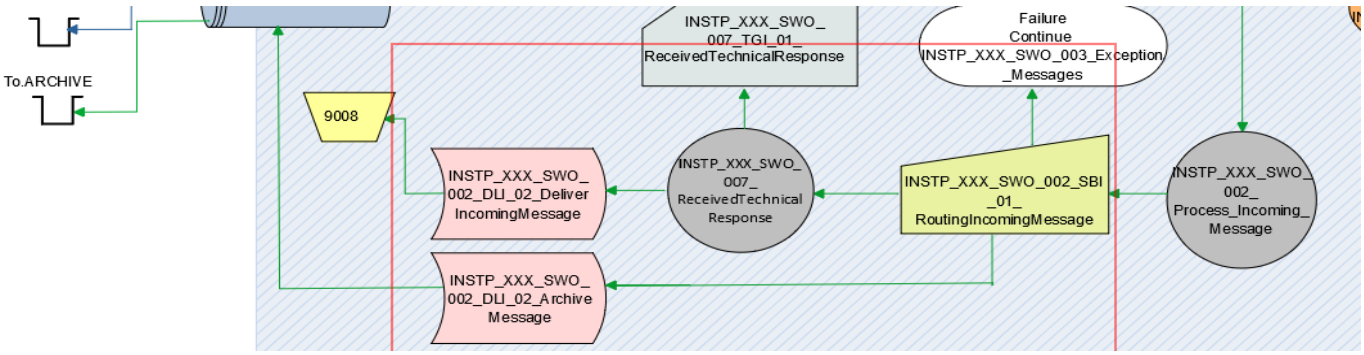
8.4.3 Analysis 1

Analysis 1 provides a C-like programming language, which can be used to analyze 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 whether 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:

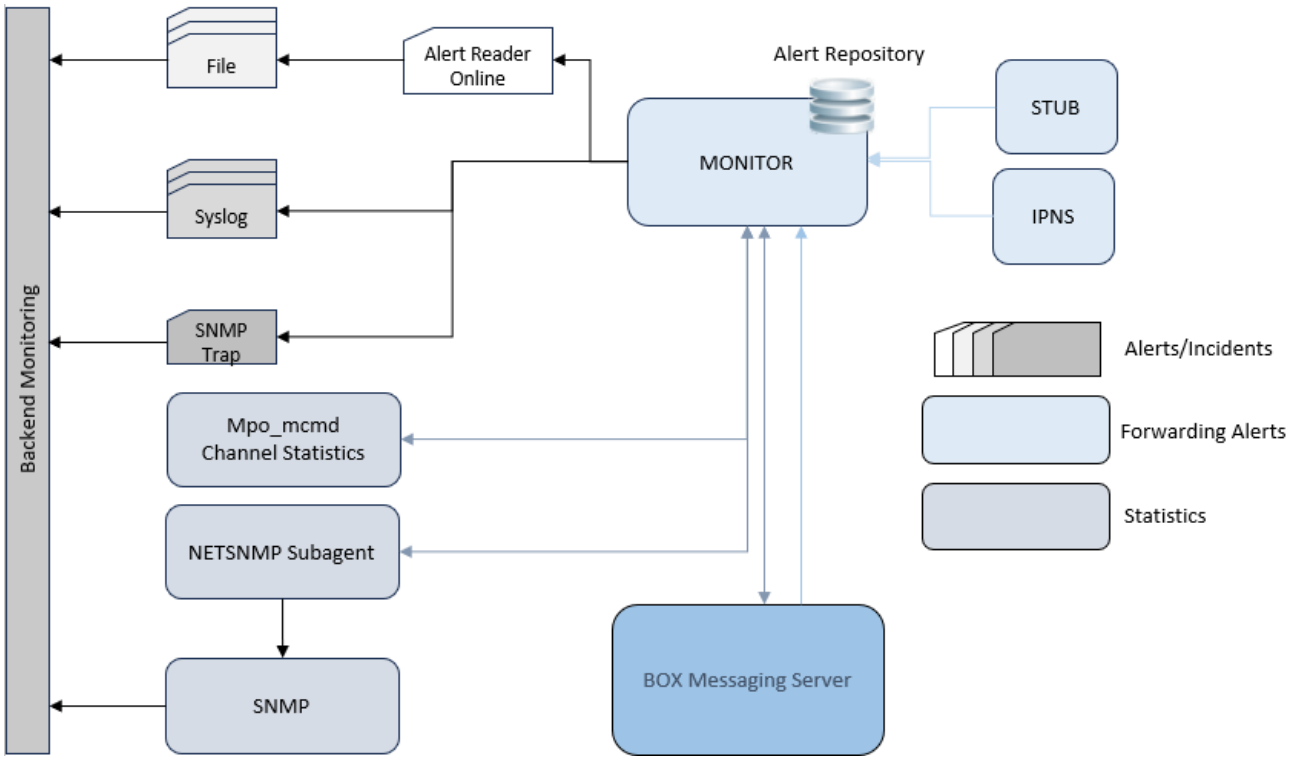
```
INSTP_XXX_SWO_002_Process_Incoming_Message

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;
```



Picture 49 Excerpt Workflow containing Analysis 1

9 Monitoring BOX with SNMP Dashboard

9.1 Monitoring BOX in non-persistence mode



Picture 50 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, the System Monitor automatically restarts it. When a domain shuts down 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 analyzed 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	

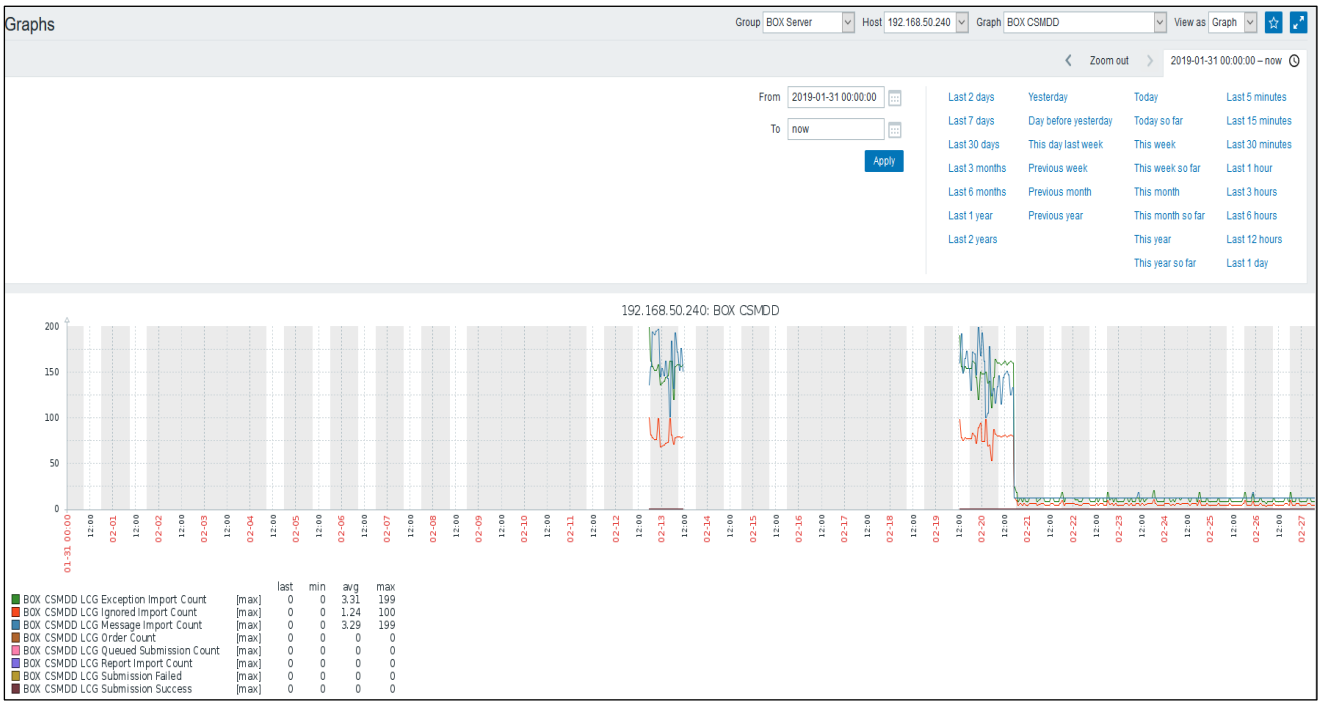
9.1.3 SNMP Integration with Zabbix

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.



Picture 51 BOX Server Health Monitoring Example

10 Appendix

10.1 Table of Graphs

10.2 Parameter

10.2.1 Analysis1

The following aliases have been defined in Analysis1 to retrieve SWIFT instant payment related data. For completeness and definition, please refer to the respective documentation, `box_analysis1_<version>.pdf`.

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

CV_OR_PROTREP_IPRTSW_MSG_REF
CV_OR_PROTREP_IPRTSW_ADDITIONAL_INFO
CV_OR_PROTREP_IPRTSW_REQUESTOR_DN
CV_OR_PROTREP_IPRTSW_RESPONDER_DN
CV_OR_PROTREP_IPRTSW_SERVICE_NAME
CV_OR_PROTREP_IPRTSW_MSG_TYPE
CV_OR_PROTREP_IPRTSW_MSG_NETWORK_REF
CV_OR_PROTREP_IPRTSW_CHANNEL_NAME
CV_OR_PROTREP_IPRTSW_PROTOCOL_CODE
CV_OR_PROTREP_IPRTSW_DEVICE_CODE
CV_OR_PROTREP_IPRTSW_POSSIBLE_DUPLICATE
CV_OR_PROTREP_IPRTSW_SEND_TIME
CV_OR_PROTREP_IPRTSW_RECEIVE_TIME
CV_OR_PROTREP_IPRTSIC_SENDER_PARTICIPANT_ID
CV_OR_PROTREP_IPRTSIC_PROTOCOL_CORRELATION_ID
CV_OR_PROTREP_IPRTSIC_BUSINESS_SERVICE_ID
CV_OR_PROTREP_IPRTSIC_SENDER_INTERFACE_ID
CV_OR_PROTREP_IPRTSIC_RECEIVER_INTERFACE_ID
CV_OR_PROTREP_IPRTSIC_PAYLOAD_MESSAGE_TYPE
CV_OR_PROTREP_IPRTSIC_SESSION_ID
CV_OR_PROTREP_IPRTSIC_CHANNEL_NAME
CV_OR_PROTREP_IPRTSIC_PROTOCOL_CODE
CV_OR_PROTREP_IPRTSIC_DEVICE_CODE
CV_OR_PROTREP_IPRTSIC_BUSINESS_VERSION
CV_OR_PROTREP_IPRTSIC_MQ_BACKOUT_COUNT
CV_OR_PROTREP_IPRTSIC_LOCAL_RECEPTION_TIME
CV_OR_PROTREP_IPRTSIC_RECEIVED_SIGNATURE_TIME
CV_OR_PROTREP_DEST_IPRTSIC_RECEIVER_PARTICIPANT_ID

SDA_PROTREP_IPRTSW_MSG_REF
SDA_PROTREP_IPRTSW_ADDITIONAL_INFO
SDA_PROTREP_IPRTSW_PRIMITIVE_ERROR_TEXT
SDA_PROTREP_IPRTSW_REQUESTOR_DN
SDA_PROTREP_IPRTSW_RESPONDER_DN
SDA_PROTREP_IPRTSW_SERVICE_NAME
SDA_PROTREP_IPRTSW_MSG_TYPE
SDA_PROTREP_IPRTSW_MSG_NETWORK_REF
SDA_PROTREP_IPRTSW_CHANNEL_NAME
SDA_PROTREP_IPRTSW_REPORT_SOURCE
SDA_PROTREP_IPRTSW_PROTOCOL_CODE
SDA_PROTREP_IPRTSW_DEVICE_CODE
SDA_PROTREP_IPRTSW_PRIMITIVE_RETURN_CODE
SDA_PROTREP_IPRTSW_POSSIBLE_DUPLICATE
SDA_PROTREP_IPRTSW_SEND_TIME
SDA_PROTREP_IPRTSW_RECEIVE_TIME
SDA_PROTREP_IPRTSIC_RECEIVER_PARTICIPANT_ID
SDA_PROTREP_IPRTSIC_SENDER_PARTICIPANT_ID
SDA_PROTREP_IPRTSIC_PROTOCOL_CORRELATION_ID
SDA_PROTREP_IPRTSIC_BUSINESS_CORRELATION_ID
SDA_PROTREP_IPRTSIC_BUSNINESS_SERVICE_ID
SDA_PROTREP_IPRTSIC_SENDER_INTERFACE_ID
SDA_PROTREP_IPRTSIC_RECEIVER_INTERFACE_ID
SDA_PROTREP_IPRTSIC_PAYLOAD_MESSAGE_TYPE
SDA_PROTREP_IPRTSIC_SESSION_ID
SDA_PROTREP_IPRTSIC_REPORT_SOURCE
SDA_PROTREP_IPRTSIC_CHANNEL_NAME
SDA_PROTREP_IPRTSIC_PROTOCOL_CODE
SDA_PROTREP_IPRTSIC_DEVICE_CODE
SDA_PROTREP_IPRTSIC_LOCAL_SUBMISSION_TIME
ISDREP_PROTREP_IPRTSW_MSG_REF
ISDREP_PROTREP_IPRTSW_ADDITIONAL_INFO
ISDREP_PROTREP_IPRTSW_PRIMITIVE_ERROR_TEXT
ISDREP_PROTREP_IPRTSW_REQUESTOR_DN
ISDREP_PROTREP_IPRTSW_RESPONDER_DN
ISDREP_PROTREP_IPRTSW_SERVICE_NAME
ISDREP_PROTREP_IPRTSW_MSG_TYPE
ISDREP_PROTREP_IPRTSW_MSG_NETWORK_REF
ISDREP_PROTREP_IPRTSW_CHANNEL_NAME
ISDREP_PROTREP_IPRTSW_REPORT_SOURCE
ISDREP_PROTREP_IPRTSW_PROTOCOL_CODE
ISDREP_PROTREP_IPRTSW_DEVICE_CODE
ISDREP_PROTREP_IPRTSW_PRIMITIVE_RETURN_CODE
ISDREP_PROTREP_IPRTSW_POSSIBLE_DUPLICATE
ISDREP_PROTREP_IPRTSW_SEND_TIME
ISDREP_PROTREP_IPRTSW_RECEIVE_TIME
ISDREP_PROTREP_IPRTSIC_RECEIVER_PARTICIPANT_ID
ISDREP_PROTREP_IPRTSIC_SENDER_PARTICIPANT_ID
ISDREP_PROTREP_IPRTSIC_PROTOCOL_CORRELATION_ID
ISDREP_PROTREP_IPRTSIC_BUSINESS_CORRELATION_ID
ISDREP_PROTREP_IPRTSIC_BUSINESS_SERVICE_ID
ISDREP_PROTREP_IPRTSIC_SENDER_INTERFACE_ID
ISDREP_PROTREP_IPRTSIC_RECEIVER_INTERFACE_ID
ISDREP_PROTREP_IPRTSIC_PAYLOAD_MESSAGE_TYPE
ISDREP_PROTREP_IPRTSIC_SESSION_ID
ISDREP_PROTREP_IPRTSIC_REPORT_SOURCE
ISDREP_PROTREP_IPRTSIC_CHANNEL_NAME
ISDREP_PROTREP_IPRTSIC_PROTOCOL_CODE
ISDREP_PROTREP_IPRTSIC_DEVICE_CODE
ISDREP_PROTREP_IPRTSIC_LOCAL_SUBMISSION_TIME
ISDREP_PROTREP_IPRTSIC_ACKNOWLEDGEMENT_TIME
ISDREP_PROTREP_IPRTSIC_RECEIVER_PARTICIPANT_ID

ISDREP_PROTREP_IPRTSIC_SENDER_PARTICIPANT_ID
ISDREP_PROTREP_IPRTSIC_PROTOCOL_CORRELATION_ID
ISDREP_PROTREP_IPRTSIC_BUSINESS_CORRELATION_ID
ISDREP_PROTREP_IPRTSIC_BUSINESS_SERVICE_ID
ISDREP_PROTREP_IPRTSIC_SENDER_INTERFACE_ID
UPMADR_IPRTSW_RESPONDER_DN
ABRECADR_IPRTSW_RESPONDER_DN
UPMADR_IPRTSIC_RECEIVER_PARTICIPANT_ID
ABRECADR_IPRTSIC_RECEIVER_PARTICIPANT_ID

Table 10.2-VIII Analysis 1 Aliases

MQ BACKOUT COUNT

Address Type	Parameter
MQ	CV_OR_PROTREP_MQ_BACKOUT_COUNT
IPRT_EB	CV_OR_PROTREP_IPRTEB_MQ_BACKOUT_COUNT

Table 10.2-IX 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