avaloq
An NEC Company

SWIFT / SIC / PostFinance / SECOM / SEPA 2022

Customization Upgrade Guide
Release Independent
Area: Settlement

Latest version of this document

The latest version of this document can be found at https://docs.avaloq.com

Feedback

Please send any feedback to documentation@avaloq.com

Avaloq Group Ltd. Allmendstr. 140 ¦ CH-8027 Zürich ¦ Switzerland

Version History

Version / Date	Section	Description of the change
Release Independent v0 / 11 October 2022	"SIC Changes for November 2022" on page 15	In **Versioned CSS** section, updated affected function name for the MX_CAMT019N_SIC_IN_CSSsource.
Release Independent v0 / 29 August 2022		This is a new document.

Contents

1 Introduction

This section describes the required changes to prepare the upgrade of SWIFT, SIC, PostFinance, SECOM and SEPA in November 2022.

Further Reading

For more information, read the following:

SWIFT Release 2022 - Business User Guide (doc. ID: 1420)	Provides information about the SWIFT Standards MT release in November 2022.
SIC Release 4.9 - November 2022 - Business User Guide (doc. ID: 2140)	Provides an overview of the changes in Avaloq Core regarding SIC Release 4.9.
SEPA Release 2022 - Business User Guide (doc. ID: 1422)	Provides an overview of the changes in Avaloq Core regarding the SEPA update that takes effect on 21 November 2022.

Table 1: Further reading

2 Customization Upgrade November 2022

2.1 Checklist for November 2022

This section serves as an overview for necessary upgrade steps in November 2022.

SWIFT

The table below provides a checklist of steps to be done before or after an upgrade.

Upgrade Step	Done
Before upgrade	
On the productive Avaloq Core, on the date the release upgrade is productive on the SWIFT FIN network, set the `netw_release_id` item of the `avq.msg.swift` base parameter to "swift$sss22".	[ ]
Check / adapt your CSS in accordance with the changes to the SWIFT messages listed in "Upgrading SWIFT MT for November 2022" on page 10. For more information, see *SWIFT Release 2022 - Business User Guide (doc. ID: 1420)*.	[ ]
After upgrade	
Upload the Bank Directory Plus directory.	[ ]

Table 2: Migration checklist SWIFT (November 2022)

SIC

The table below provides a checklist of steps to be done before or after an upgrade.

Upgrade Step	Done
Before upgrade	
On the productive Avaloq Core, on the date the release upgrade is productive on the SIC network, set the `netw_release_id` item of the `avq.msg.swift` base parameter to "swift$sss22".	[ ]
Copy CSS from old version to new version (and adapt as required) and resolve any data dictionary conflicts.	[ ]
Ensure that any underlying procedures or functions are compatible with both the old and new versions.	[ ]
To assist with & handle path differences, there are message version agnostic functions/procedures in MX_UTIL. If not using these, a cast “with mem_msg_mx_pacs008n_v08/mem_msg_mx_pacs008n (i_msg) as m do” around any fields that are different between versions can be used to run correctly without runtime errrors.	[ ]

Table 3: Migration checklist SIC (November 2022)

SEPA

The table below provides a checklist of steps to be done before or after an upgrade.

Upgrade Step	Done
Before upgrade	
On the productive Avaloq Core, on the date the release upgrade is productive on the SEPA network, set the `netw_release_id` item of the `avq.msg.swift` base parameter to "swift$sss22".	[ ]

Table 4: Migration checklist SEPA (November 2022)

3 SWIFT MT Customization Upgrade November 2022

3.1 SWIFT MT Changes for November 2022

The following sections describe the changes you must make to your SWIFT MT implementation for the SWIFT MT 2022 release.

For more information about kernel changes made to comply with the latest SWIFT specification, see *SWIFT Release 2022 - Business User Guide (doc. ID: 1420)*.

3.1.1 SWIFT MT Modified Avaloq Sources November 2022

The following sources were modified:

Construct	Description
Network Structure: SWIFT3	Name and format of field **35C** changed to Digital Token Identifier and `8x[/30x]`. Added field **36D** with the format `4!c//4!c/30d`.
Network Structure: SWIFT9	Added field **93E** with the format `4!c/4!c/4!c/[N]30d`. Added field **93F** with the format `4!c/[8c]/4!c/[N]30d`. Added field **97D** with the format `4!c/[8c]/140x`.

Table 5: Modified Avaloq constructs for SWIFT November 2022

3.2 Upgrading SWIFT MT for November 2022

To keep up to date with the SWIFT MT 2022 release upgrade, complete the following steps.

The `netw_release_id` item of the `avq.msg.swift` base parameter defines the currently valid release for SWIFT, SIC, SEPA and PostFinance. This item must be set at the date of a release upgrade. If this base parameter is not set, or is set with the value from a previous release, the implementation assumes the lowest possible or previous release for a given stream.

The `netw_release_id` base parameter item of the `avq.msg.sic` base parameter is no longer used and does not need to be updated. Instead, the `netw_release_id` item of the `avq.msg.swift` base parameter is used for all S4 and PostFinance releases.

The changes for the 2022 release can be installed *before* the actual release date.

Message structures are extended to support the old and new releases. Outgoing message handlers are capable of completing the messages in both the old *and* new formats for each network.

The decision of which version to use is specified in the `netw_release_id` item of the `avq.msg.swift` base parameter.

3.2.1 Prerequisite

Before testing the changes delivered for SWIFT Standards MT Release November 2022, you must set the `netw_release_id` item of the `avq.msg.swift` base parameter to "swift$sss22" on the testing environment. When moving the change to production, you must also ensure that this value is set.

3.2.2 Steps to Follow

1. On the productive Avaloq Core, on the date the release upgrade is productive on the SWIFT FIN network, set the `netw_release_id` item of the `avq.msg.swift` base parameter to "swift$sss22".
2. Check your CSS for the following messages types. Some fields and options have been removed or replaced by new fields and options. For information about the changes that have been delivered, ensure you read *SWIFT Release 2022 - Business User Guide (doc. ID: 1420)*.
   - MT300 Foreign Exchange Confirmation
   - MT304 Advice/Instruction of a Third Party Deal
   - MT305 Foreign Currency Option Confirmation
   - MT306 Foreign Currency Option Confirmation
   - MT340 Forward Rate Agreement Confirmation
   - MT341 Forward Rate Agreement Settlement Confirmation
   - MT360 Single Currency Interest Rate Derivative Confirmation
   - MT361 Cross Currency Interest Rate Swap Confirmation
   - MT500 Instruction to Register
   - MT501 Confirmation of Registration or Modification
   - MT502 Order to Buy or Sell
   - MT508 Intra-Position Advice

- MT509 Trade Status Message
- MT510 Registration Status and Processing Advice
- MT513 Client Advice of Execution
- MT514 Trade Allocation Instruction
- MT515 Client Confirmation of Purchase or Sale
- MT518 Market-Side Securities Trade Confirmation
- MT519 Modification of Client Details
- MT524 Intra-Position Instruction
- MT527 Triparty Collateral Instruction
- MT530 Transaction Processing Command
- MT535 Statement of Holdings
- MT536 Statement of Transactions
- MT537 Statement of Pending Transactions
- MT538 Statement of Intra-Position Advice
- MT540 Receive Free
- MT541 Receive against Payment
- MT542 Deliver Free
- MT543 Deliver against Payment
- MT544 Receive Free Confirmation
- MT545 Receive against Payment Confirmation
- MT546 Deliver Free Confirmation
- MT547 Deliver against Payment Confirmation
- MT548 Settlement Status and Processing Advice
- MT549 Request for Statement / Status Advice
- MT558 Triparty Collateral Status and Processing Advice
- MT564 Corporate Action Notification
- MT565 Corporate Action Instruction
- MT566 Corporate Action Confirmation
- MT567 Corporate Action Status and Processing Advice
- MT568 Corporate Action Narrative
- MT569 Triparty Collateral and Exposure Statement
- MT575 Report of Combined Activity
- MT576 Statement of Open Orders
- MT578 Settlement Allegement
- MT586 Statement of Settlement Allegements
- MT600 Commodity Trade Confirmation
- MT601 Commodity Option Confirmation

Tags modified from a list to a list of options

Some tags are modified from a list to a list of options with SWIFT Release 2022.

An example of this change is tag 36, which was previously defined as 36B, and becomes 36a with options 36B and 36D with the SWIFT Release 2022.

This change has the following consequences:

In the previous release, the tag was accessible directly by calling the field. For example:

```
i_m.ordrdet.qty_instru_list.qty_instru.qual = 'ORDR'
```

With SWIFT Release 2022, the name of the compound has changed and the tag must be called with the option. For example:

```
i_m.ordrdet.qty_instru_list_opt.qty_instru.b.qual = 'ORDR'
```

If the affected tags are accessed in your customization, you need to adapt it to reflect this change.

Kernel logic will ensure that:

- If the change related to the SWIFT Release 2022 is installed, and an older previous network release is defined: the tag is filled with / accessed at its previous location.
- If the change related to the SWIFT Release 2022 is installed, and the network release SSS22 is defined: the tag is filled with / accessed at its new location.

This change affects the following messages:

- MT500 – Instruction to Register
- MT501 – Confirmation of Registration or Modification
- MT502 – Order to Buy or Sell
- MT508 – Intra-Position Advice
- MT509 – Trade Status Message
- MT510 – Collateral Claim
- MT513 – Client Advice of Execution
- MT514 – Trade Allocation Instruction
- MT515 – Client Confirmation of Purchase or Sale
- MT518 – Market-Side Securities Trade Confirmation
- MT519 – Modification of Client Details
- MT524 – Intra-Position Instruction
- MT527 – Triparty Collateral Instruction
- MT530 – Transaction Processing Command
- MT535 – Statement of Holdings
- MT536 – Statement of Transaction
- MT537 – Statement of Pending Transactions

- MT538 – Statement of Intra-Position Advices
- MT540 – Receive Free
- MT541 – Receive against Payment
- MT542 – Deliver Free
- MT543 – Deliver against Payment
- MT544 – Receive Free Confirmation
- MT545 – Receive against Payment Confirmation
- MT546 – Deliver Free Confirmation
- MT547 – Deliver against Payment Confirmation
- MT548 – Settlement Status and Processing Advice
- MT549 – Request for Statement / Status Advice
- MT558 – Triparty Collateral Status and Processing Advice
- MT564 – Corporate Action Notification
- MT565 – Corporate Action Instruction
- MT566 – Corporate Action Confirmation
- MT567 – Corporate Action Status and Processing Advice
- MT568 – Corporate Action Narrative
- MT569 – Triparty Collateral and Exposure Statement
- MT575 – Report of Combined Activity
- MT576 – Statement of Open Orders
- MT578 – Settlement Allegement
- MT586 – Statement of Settlement Allegements

3.2.3 Result

The SWIFT MT implementation is up to date.

4 PostFinance Customization Upgrade November 2022

4.1 PostFinance Changes for November 2022

No kernel changes are delivered for the PostFinance network. To keep your system current for the PostFinance November 2022 update, review the official specification and modify the relevant customization sources if required.

5 SIC Customization Upgrade November 2022

5.1 SIC Changes for November 2022

For information about the changes that Avaloq has implemented in kernel code to comply with the latest SIC specifications, see *SIC Release 4.9 - November 2022 - Business User Guide (doc. ID: 2140)*.

5.2 SIC Changes for November 2022

The information below describes the changes you need to make to your SIC/euroSIC implementation for the SIC 4.9 release in November 2022. These updates are required due to changes to the kernel code implemented by Avaloq, in accordance with the latest SIC specifications. Review any existing customization for the related message types to ensure that your implementation is valid for the 2022 specification.

For more information about about kernel changes made to comply with the latest SIC specification, see *SIC Release 4.9 - November 2022 - Business User Guide* (doc. ID: 2140).

Approval of the "Settlement Time Request" Element

With effect from Release 4.9 on 18 November 2022, the 'Settlement Time Request' element is allowed for the CSTPMT (pacs.008), F2FPMT and COVPMT (both pacs.009) payment types.

As the SttlmTmReq element is optional and must be bilaterally agreed for the above payment types, Avaloq has made no changes.

Discontinuation of the Delivery of Bank Master Data via File Transfer

The bank master is currently provided via the SIC Ltd website as a manual download, via the file transfer service of SIX and, since 5 June 2020, also via REST API (JSON format). Bank master delivery via file transfer is therefore no longer necessary. With effect from Release 4.9 on 18 November 2022, bank master delivery via file transfer will be discontinued.

As the other options are still supported, Avaloq has made no changes.

Removal of the "TCMSTM" Payment Type from various messages

The third-party debit system currently maintains the following payment types in the SIC system for clearing corresponding net positions using third-party system payments (pacs.009):

- EFT/POS settlement (POSSTM)
- Bancomat settlement (BCMSTM)
- Tancomat settlement (TCMSTM)

Due to technical consolidations within debit card processing, the explicit separation of payments from "Tancomat settlement" no longer meets current needs. For these reasons and the resulting low transaction volumes of the payment type, a dedicated payment type in the SIC system is no longer justified.

Therefore, with effect from Release 4.9 on 18 November 2022, the 'TCMSTM' payment type will no longer be supported in the SIC system and will therefore also be removed from all affected message definitions.

Avaloq has made no changes, as no specific Tancomat processing is implemented in kernel.

Mandatory Use of UETR in Customer Payments (pacs.008) and Bank and Third-Party System Payments (pacs.009)

Due to developments in the international payment traffic environment, it is recognised that the use of 'UETR' in the interbank area is developing in the direction of a globally standardized and clear common practice. This circumstance is only partially taken into account in the SIC and euroSIC system today, as the "UETR" is not yet mandatory in all payment types for customer payments (pacs.008) as well as bank and third-party system payments (pacs.009).

With effect from Release 4.9 on 18 November 2022, the new dedicated "UETR" XML element in customer payments (pacs.008) and bank and third-party system payments (pacs.009) becomes mandatory for all payment types.

Avaloq has made modifications to the logic to ensure that the new element is filled for the updated message versions.

Removal of the "ISR payment" Payment Type and the specification of the "IS reference number" Account in Customer Payments (pacs.008)

The ISR payment type (pacs.008/ESRPMT) is obsolete due to the discontinuation of the ISR slip. The IS reference number (PSREF) as a possible expression of a creditor account in the "General customer payment" (pacs.008/CSTPMT) payment type is also therefore superfluous due to the discontinuation of the IS slip.

As a result, with effect from Release 4.9 on 18 November 2022, The 'ISR payment' payment type (pacs.008/ESRPMT) and the reference number (PSREF) as a possible expression of a creditor account in the 'General customer payment' (pacs.008/CSTPMT) payment type will be removed.

Avaloq has made modifications to the logic in the relevant sources where required.

Update of ISO 20022 Messages to ISO Version Level 2019

The ISO 20022 messages currently used in the SIC and euroSIC system correspond to the ISO version level from 2009. With effect from Release 4.9 on 18 November 2022, new message versions with version level 2019 will be introduced for all affected ISO 20022 messages. In addition, modifications will be made to harmonize the local ISO 20022 interbank message definitions with Swiss Payment Standards (customer-bank messages) as well as international market practices (e.g. CBPR+, HVPS+, TARGET2).

The governing interbank committee has also decided to extend the permitted character set in SIC/euroSIC within this change request and to align it with the scope of the "Swiss Payments Standards 2022" (SPS 2022). This will enable SIC/euroSIC participants, for example, to transmit any umlauts in text elements of a customer order unchanged via SIC/euroSIC in the future and to present them unchanged to the creditor in a credit note or account statement.

Avaloq has made modifications where required to support the new message versions.

Introduction of Mandatory Return of Bank Payments (pacs.009) via pacs.004

If an incoming 'Bank and third-party system payments' (pacs.009) payment cannot be processed, the return is currently made via pacs.009. In contrast, the return for incoming 'customer payments' (pacs.008) payments must always be made via pacs.004.

With effect from Release 4.9 on 18 November 2022, returns from incoming payments of the "FI-to-FI payment" (F2FPMT), "Compensation payment" (CMPPMT) and "Cover payment" (COVPMT) payment types will be made using the pacs.004 message.

Avaloq has made modifications where required to support the new use case.

Introduction of Return Requests (camt.056) and Rejection of Return Requests (camt.029) in the euroSIC System

Due to modifications in the TARGET2 system of the European Central Bank, a return request or its rejection will also be introduced on an international level for non-SEPA payments as of November 2022. In addition, harmonization between the SIC and euroSIC systems is necessary.

Therefore, with effect from Release 4.9 on 18 November 2022, the 'Return request' (camt.056) and 'Return request rejection' (camt.029) use cases are introduced in the euroSIC system.

Avaloq has made modifications where required to support the new use case.

5.2.1 MX_UTIL Public Functions

To assist with path differences, functions and procedures are provoded in MX_UTIL which work regardless of message version, and to handle these differences. For further guidance on this, see Upgrading SIC for November 2022. The functions and procedures as listed below:

Functions

BIC/BICFI

Name: fin_instn#bic(i_node)

Description: Get <BIC> or <BICFI> value from <FinInstId>

Supports: PACS.002/004/008/009 and CAMT.029/056

Example:

```
mx_util.fin_instn#bic(i_msg.document.fi_to_fi_cstmr_cdt_trf.cdt_trf_tx_inf_list(1).cdtr_agt.fin_instn_
id);
```

BICOrBEI/AnyBIC

Name: org#bic(i_node)

Description: Get <BICOrBEI> or <AnyBIC> value from <FinInstId>

Supports: PACS.002/004/008/009 and CAMT.029/056

Example:

```
mx_util.org#bic(i_msg.document.fi_to_fi_cstmr_cdt_trf.cdt_trf_tx_inf_list(1).dbtr .id_v.opt.org_id);
```

SvcLvl/SvcLvlList

Name: pmt_tp_inf#svc_lvl_cd(i_node)

Description: Get <Cd> value from <SvcLvl>

Supports: PACS.004/008/009 and CAMT.029/056

Example:

```
mx_util.pmt_tp_inf#svc_lvl_cd(i_msg.document.fi_to_fi_cstmr_cdt_trf.cdt_trf_tx_inf_list(1).pmt_tp_inf);
```

Name: pmt_tp_inf#svc_lvl_prtry (i_node)

Description: Get <Prtry> value from <SvcLvl>

Supports: PACS.004/008/009 and CAMT.029/056

Example:

```
mx_util.pmt_tp_inf#svc_lvl_prtry(i_msg.document.fi_to_fi_cstmr_cdt_trf.cdt_trf_tx_inf_list(1).pmt_tp_
inf);
```

OrgnlMsgId

Name: tx_inf_and_sts#orgnl_msg_id (i_msg)

Description: Get <OrgnlMsgId> value from <OrgnlGrpInfAndSts> or <TxInfAndSts>

Supports: PACS.002

Example:

```
mx_util.tx_inf_and_sts#orgnl_msg_id(i_msg => i_msg);
```

CdtDbtInd

Name: ttl_ntries#cdt_dbt_ind(i_node)

Description: Get <CdtDbtInd> or <TtlNetNtry><CdtDbtInd> value from <TtlNetNtry>

Supports: CAMT.052

Example:

```
mx_util.ttl_ntries#cdt_dbt_ind(i_node => i_msg.rpt.txs_summry.ttl_ntries_per_bk_tx_cd_list(1));
```

Amt

Name: ttl_ntries#amt (i_node)

Description: Get <Sum> or <TtlNetNtry><Amt> value from <TtlNtriesPerBkTxCd>

Supports: CAMT.052

Example:

```
mx_util.ttl_ntries#amt(i_node = > i_msg.rpt.txs_summry.ttl_ntries_per_bk_tx_cd_list(1));
```

BICOrBEI/AnyBIC/BICFI

Name: prty#bic(i_node)

Description: Get <BICOrBEI> value for old XSD, get <Pty><AnyBIC> if new XSD is with PACS.008 or get <Agt><BICFI> if new XSD is with PACS.009 from <Dbtr>/<Cdtr>/<UltmtDbtr>/<UltmtCdtr>

Supports: PACS.004 and CAMT.056

Example:

```
mx_util.prty#bic(i_node => i_msg.document.fi_to_fi_pmt_cxl_req.undrlyg.tx_inf.orgnl_tx_ref.cdtr);
```

Pty

Name: msg#prty(i_node)

Description: Get <Pty> node if new XSD from <Dbtr>/<Cdtr>/<UltmtDbtr>/<UltmtCdtr> else this node.

Supports: PACS.004 and CAMT.056

Example:

```
mx_util.msg#prty(i_node => i_msg.document.fi_to_fi_pmt_cxl_req.undrlyg.tx_inf.orgnl_tx_ref.cdtr);
```

Nm

Name: pty#nm(i_node)

Description: Get <Nm> vale from <Dbtr>/<Cdtr>/<UltmtDbtr>/<UltmtCdtr>.

Supports: PACS.004 and CAMT.056

Example:

```
mx_util.pty#nm(i_node => i_msg.document.fi_to_fi_pmt_cxl_req.undrlyg.tx_inf.orgnl_tx_ref.cdtr);
```

AddrLine

Name: prty#addr_line_list_cnt(i_node)

Description: Count <AddrLine> from <Dbtr>/<Cdtr>/<UltmtDbtr>/<UltmtCdtr>.

Supports: PACS.004 and CAMT.056

Example:

```
mx_util.prty#addr_line_list_cnt(i_node =>
         i_msg.document.fi_to_fi_pmt_cxl_req.undrlyg.tx_inf.orgnl_tx_ref.cdtr);
```

Name: prty#unstruct_addr(i_node)

Description: Get unstructured address from <Dbtr>/<Cdtr>/<UltmtDbtr>/<UltmtCdtr>.

Supports: PACS.004 and CAMT.056

Example:

```
>mx_util.prty# unstruct_addr(i_node => i_msg.document.fi_to_fi_pmt_cxl_req.undrlyg.tx_inf.orgnl_tx_
ref.cdtr);
```

Name: prty#addr_line (i_node, i_seq_nr)

Description: Get <AdrLine> value from <Dbtr>/<Cdtr>/<UltmtDbtr>/<UltmtCdtr> with specific sequence number.

Supports: PACS.004 and CAMT.056

Example:

```
>mx_util.prty#addr_line(i_node => i_msg.document.fi_to_fi_pmt_cxl_req.undrlyg.tx_inf.orgnl_tx_
ref.cdtr);
```

PstlAdr

Name: prty#pstl_adr(i_node)

Description: Get <PstlAdr> node from <Dbtr>/<Cdtr>/<UltmtDbtr>/<UltmtCdtr>.

Supports: PACS.004

Example:

```
mx_util.prty#pstl_adr(i_node => i_msg.document.fi_to_fi_pmt_cxl_req.undrlyg.tx_inf.orgnl_tx_ref.cdtr);
```

FinInstnId

Name: msg#instd_agt(i_msg)

Description: Get <FinInstnId> node from <InstdAgt>.

Supports: PACS.004/008/009

Example: mx_util.msg#instd_agt(i_msg);

Name: msg#instg_agt(i_msg)

Description: Get <FinInstnId> node from <InstgAgt>.

Supports: PACS.004/008/009

Example:

```
mx_util.msg#instg_agt(i_msg);
```

IntrBkSttlmDt

Name: msg#intr_bk_sttlm_dt(i_msg)

Description: Get date from <IntrBkSttlmDt>

Supports: PACS.004/008/009

Example:

```
mx_util.msg#intr_bk_sttlm_dt(i_msg => msg);
```

MmbId

Name: fin_instn_id#mmb_id(i_node)

Description: Get <MmbId> value from <FinInstnId>. This can be used together with msg#instd_agt(i_msg) or with msg#instg_agt(i_msg) to get its <MmbId> value.

Supports: PACS.004

Example:

```
mx_util.fin_instn_id#mmb_id(mx_util.msg#instd_agt(msg));
```

ClrSysId

Name: fin_instn_id#clr_sys_id(i_node)

Description: Get <ClrSysId> value from <FinInstnId>. This can be used together with msg#instd_agt(i_msg) or

with msg#instg_agt(i_msg) to get its < ClrSysId > value.

Supports: PACS.004

Example:

```
mx_util.fin_instn_id#clr_sys_id(mx_util.msg#instd_agt(msg));
```

Id

Name: fin_instn_id#id_v(i_node)

Description: Get <Id> value from <FinInstnId>. This can be used together with msg#instd_agt(i_msg) or with msg#instg_agt(i_msg) to get its <Id> value.

Supports: PACS.004

Example:

```
mx_util.fin_instn_id#id_v(mx_util.msg#instd_agt(msg));
```

Nm

Name: fin_instn_id#nm(i_node)

Description: Get <Nm> value from <FinInstnId>. This can be used together with msg#instd_agt(i_msg) or

with msg#instg_agt(i_msg) to get its <Nm> value.

Supports: PACS.004

Example:

```
mx_util.fin_instn_id#nm(mx_util.msg#instd_agt(msg));
```

PstlAdr

Name: fin_instn_id#pstl_adr(i_node)

Description: Get <PstlAdr> value from <FinInstnId>. This can be used together with msg#instd_agt(i_msg) or

with msg#instg_agt(i_msg) to get its <PstlAdr> value.

Supports: PACS.004

Example:

```
mx_util.fin_instn_id#pstl_adr(mx_util.msg#instd_agt(msg));
```

ClrSys

Name: sttlm_inf#clr_sys (i_node)

Description: Get <Cd> or <Prtry> value from <SttlmInf><ClrSys>

Supports: PACS.004

Example:

```
mx_util.sttlm_inf#clr_sys(i_msg.document.pmt_rtr.grp.grp_hdr.sttlm_inf);
```

FiCdtTrf

Name: document#fi_cdt_trf(i_node)

Description: Get <FiCdtTrf> node from <Document>

Supports: PACS.009

Example:

```
mx_util.document#fi_cdt_trf(i_msg.document);
```

Procedures

BIC/BICFI

Name: fin_instn#set_bic (i_node, i_bic)

Description: Set <BIC> or <BICFI> value on <FinInstnId>

Supports: PACS.002/004/008/009 and CAMT.029/056

Example:

```
mx_util.fin_instn#set_bic (
  i_node => i_msg.document.fi_to_fi_cstmr_cdt_trf.cdt_trf_tx_inf_list(1).dbtr.id_v.opt.org_id
  ,i_bic => 'testbic' );
```

BICOrBEI/AnyBIC

Name: org#set_bic(i_node, i_bic)

Description: Set <BICOrBEI> or <AnyBIC> value on <OrgId>

Supports: PACS.002/004/008/009 and CAMT.029/056

Example:

```
mx_util.org#set_bic(
  i_node => i_msg.document.fi_to_fi_cstmr_cdt_trf.cdt_trf_tx_inf_list(1).dbtr.id_v.opt.org_id
  ,i_bic => 'testbic' );
```

SvcLvl/SvcLvlList

Name: pmt_tp_inf#set_svc_lvl_cd(i_node, i_svc_lvl)

Description: Set <Cd> value on <SvcLvl>

Supports: PACS.004/008/009 and CAMT.029/056

Example:

```
mx_util.pmt_tp_inf#set_svc_lvl_cd(
  i_node => i_msg.document.fi_to_fi_cstmr_cdt_trf.cdt_trf_tx_inf_list(1).pmt_tp_inf
  ,i_svc_lvl => 'SEPA');
```

Name: pmt_tp_inf#set_svc_lvl_prtry(i_node, i_svc_lvl)

Description: Set <Prtry> value on <SvcLvl>

Supports: PACS.004/008/009 and CAMT.029/056

Example:

```
mx_util.pmt_tp_inf#set_svc_lvl_prtry(
  i_node => i_msg.document.fi_to_fi_cstmr_cdt_trf.cdt_trf_tx_inf_list(1).pmt_tp_inf
  ,i_svc_lvl => 'test');
```

ClrSys

Name: sttlm_inf#set_clr_sys(i_node, i_revs_type)

Description: Set <Cd> or <Prtry> value on <SttlmInf><ClrSys>.

Supports: PACS.004

Example:

```
mx_util.sttlm_inf#set_clr_sys(
  i_node => i_msg.document.pmt_rtr.grp.grp_hdr.sttlm_inf
  ,i_revs_type => 'SIC');
```

BICOrBEI/AnyBIC/BICFI

Name: prty#set_bic(i_node, i_orgnl_msg_nm_id, i_bic)

Description: Set <BICOrBEI> value for old XSD. Set <Pty><AnyBIC for new XSD with PACS.008 or set <Agt><BICFI> if new XSD with PACS.009.

Supports: PACS.004 and CAMT.056

Example:

```
mx_util.prty#set_bic(
  i_node => i_msg.document.fi_to_fi_pmt_cxl_req.undrlyg.tx_inf.orgnl_tx_ref.cdtr
  ,i_orgnl_msg_nm_id => 'pacs.008'
  ,i_bic => 'test');
```

Nm

Name: pty#set_nm(i_node, i_orgnl_msg_nm_id, i_nm)

Description: Set <Nm> value on <Dbtr>/<Cdtr>/<UltmtDbtr>/<UltmtCdtr>.

Supports: PACS.004 and CAMT.056

Example:

```
mx_util.pty#set_nm(
  i_node => i_msg.document.fi_to_fi_pmt_cxl_req.undrlyg.tx_inf.orgnl_tx_ref.cdtr
  ,i_orgnl_msg_nm_id => 'pacs.008'
  ,i_nm => 'test');
```

AdrLine

Name: prty#set_addr_line(i_node, i_addr)

Description: Set <AdrLine> value on <Pty><PstlAdr> or <PstlAdr>.

Supports: CAMT.056

Example:

```
mx_util.prty#set_addr_line(
  i_node => i_msg.document.fi_to_fi_pmt_cxl_req.undrlyg.tx_inf.orgnl_tx_ref.dbtr
  ,i_addr => 'test address Almendstrasse 8000 Zurich' );
  Ctry - set <ctry> value on <pstladr> or <pty><pstladr>
```

Name: prty#set_pstl_adr_ctry(i_node, i_ctry)

Description: Set `<Ctry>` value on `<Pty><PstlAdr>` or `<PstlAdr>`.

Supports: CAMT.056

Example:

```
mx_util.prty#set_pstl_adr_ctry(
  i_node => i_msg.document.fi_to_fi_pmt_cxl_req.undrlyg.tx_inf.orgnl_tx_ref.dbtr
  ,i_ctry => 'CH');
```

MmbId

Name: fin_instn_id#set_mmb_id(i_node, i_mm_id)

Description: Set `<MmbId>` value on `<FinInstnId>`. This can be used together with `msg#instd_agt(i_msg)` or `msg#instg_agt(i_msg)` to set its `<MmbId>`.

Supports: PACS.004

Example:

```
mx_util.fin_instn_id#set_mmb_id(i_node => mx_util.msg#instd_agt(msg), i_mmb_id => 'test' );
```

ClrSysId

Name: fin_instn_id#set_clr_sys_id(i_node, i_clr_sys_id)

Description:`Set <ClrSysId><Cd>` value on `<FinInstnId>`. This can be used together with `msg#instd_agt(i_msg)` or `msg#instg_agt(i_msg)` to set its `<ClrSysId>`.

Supports: PACS.004

Example:

```
mx_util.fin_instn_id#set_clr_sys_id(i_node => mx_util.msg#instd_agt(msg), i_clr_sys_id => 'CHSIC'
```

5.2.2 Versioned CSS

Once you have activated the SIC November 2022 release (by setting the base parameter item `NETW_RELEASE_ID` to "SSS22" or greater in the `AVQ.MSG.SWIFT` base parameter), only the CSS exits suffixed with the new versions are called. This section shows the naming for the existing, unversioned (pre-November 2022) CSS procedures and functions and their versioned equivalents.

Any use of a shared procedure to handle both old and new CSS code needs to be compatible with both old and new versions, or a runtime exception can occur.

pacs.002 IN

mx_pacs002n_ogrp_sic_in_css

- msg#prc_pay_nack => msg#prc_pay_nack_v10
- msg#prc_inpay_nack => msg#prc_inpay_nack_v10
- msg#prc_settle_nack => msg#prc_settle_nack_v10

mx_pacs002n_trx_sic_in_css

- msg#prc_pay_nack => msg#prc_pay_nack_v10
- msg#prc_inpay_nack => msg#prc_inpay_nack_v10
- msg#prc_settle_nack => msg#prc_settle_nack_v10

pacs.004 OUT

mx_pacs004n_sic_out_css

- ovr_msg => ovr_msg_v09

pacs.004 IN

mx_pacs004n_trx_sic_in_css

- msg#init => msg#init_v09
- msg#is_fwd => msg#is_fwd_v09
- msg#order_type_id => msg#order_type_id_v09
- msg#paytb_bp_id => msg#paytb_bp_id_v09
- msg#final_cred_macc_id => msg#final_cred_macc_id_v09
- msg#cred_macc_id => msg#cred_macc_id_v09
- msg#deb_macc_id => msg#deb_macc_id_v09
- msg#origin_country_id => msg#origin_country_id_v09
- msg#dest_country_id => msg#dest_country_id_v09
- msg#fill_extn => msg#fill_extn_v09
- doc#valid => doc#valid_v09
- doc#proceed => doc#proceed_v09

pacs.008 OUT

mx_pacs008n_sic_out_css

- ovr_msg => ovr_msg_v08

pacs.008 IN

mx_pacs008n_trx_sic_in_css

- msg#init => msg#init_v08
- msg#is_fwd => msg#is_fwd_v08

- msg#chk_exist_doc => msg#chk_exist_doc_v08
- msg#exist_doc_id => msg#exist_doc_id_v08
- msg#order_type_id => msg#order_type_id_v08
- msg#paytb_bp_id => msg#paytb_bp_id_v08
- msg#final_cred_macc_id => msg#final_cred_macc_id_v08
- msg#cred_macc_id => msg#cred_macc_id_v08
- msg#deb_macc_id => msg#deb_macc_id_v08
- msg#origin_country_id => msg#origin_country_id_v08
- msg#dest_country_id => msg#dest_country_id_v08
- msg#add_dbtr_cost => msg#add_dbtr_cost_v08
- doc#fill_extn => doc#fill_extn_v08
- doc#valid => doc#valid_v08
- doc#proceed => doc#proceed_v08

pacs.009 OUT

mx_pacs009n_sic_f2f_out_css

- ovr_msg => ovr_msg_v08

mx_pacs009n_sic_cmp_out_css

- ovr_msg => ovr_msg_v08

mx_pacs009n_sic_cov_out_css

- ovr_msg => ovr_msg_v08

pacs.009 IN

mx_pacs009n_trx_sic_in_css

- msg#init => msg#init_v08
- msg#is_fwd => msg#is_fwd_v08
- msg#is_chqprc => msg#is_chqprc_v08
- msg#chk_exist_doc => msg#chk_exist_doc_v08
- msg#exist_doc_id => msg#exist_doc_id_v08
- msg#order_type_id => msg#order_type_id_v08
- msg#paytb_bp_id => msg#paytb_bp_id_v08
- msg#final_cred_macc_id => msg#final_cred_macc_id_v08
- msg#cred_macc_id => msg#cred_macc_id_v08
- msg#deb_macc_id => msg#deb_macc_id_v08
- msg#origin_country_id => msg#origin_country_id_v08
- msg#dest_country_id => msg#dest_country_id_v08
- msg#add_dbtr_cost => msg#add_dbtr_cost_v08

- inpay#fill_extn => doc#fill_extn_v08
- inpay#valid => inpay#valid_v08
- inpay#proceed => inpay#proceed_v08
- settle#valid => settle#valid_v08
- settle#proceed => settle#proceed_v08
- chqprc#valid => chqprc#valid_v08
- chqprc#proceed => chqprc#proceed_v08
- msg#proceed => msg#proceed_v08

camt.008 OUT

mx_camt008n_sic_out_css

- ovr_msg => ovr_msg_v08

camt.019 IN

mx_camt019n_sic_in_css

- msg#do_appy_cut_off_times => msg#do_appy_cut_off_times_v07

camt.025 IN

mx_camt025n_trx_sic_in_css

- msg#do_ovr_hdl_msg => msg#do_ovr_hdl_msg_v05
- msg#ovr_hdl_msg => msg#ovr_hdl_msg_v05
- msg#prc_pay_nack => msg#prc_pay_nack_v05
- msg#prc_inpay_nack => msg#prc_inpay_nack_v05
- msg#prc_settle_nack => msg#prc_settle_nack_v05

camt.029 IN

mx_camt029n_trx_in_css

- msg#order_type_id => msg#order_type_id_v09

camt.029 OUT

mx_camt029n_sic_bdl_out_css

- pay#ovr_assgnr => pay#ovr_assgnr_v09
- pay#ovr_assgne => pay#ovr_assgne_v09

camt.050 IN

mx_camt050n_sic_in_css

- msg#do_ovr_hdl_msg => msg#do_ovr_hdl_msg_v05
- msg#ovr_hdl_msg => msg#ovr_hdl_msg_v05

camt.050 OUT

mx_camt050n_sic_out_css

- msg#do_ovr_hdl_msg => msg#do_ovr_hdl_msg_v05
- msg#ovr_hdl_msg => msg#ovr_hdl_msg_v05

camt.052 IN

mx_camt052n_rpt_sic_in_css

- msg#is_recon_msg => msg#is_recon_msg_v08
- msg#do_gen_doc => msg#do_gen_doc_v08
- msg#ovr_hdl_msg => msg#ovr_hdl_msg_v08
- msg#cont_id => msg#cont_id_v08
- msg#deb_pos_id => msg#deb_pos_id_v08

mx_camt052n_rpt_in_recon_css

- gen_ref_1 => gen_ref_1_v08
- gen_ref_2 => gen_ref_2_v08
- gen_ref_3 => gen_ref_3_v08
- gen_ref_4 => gen_ref_4_v08
- gen_ref_5 => gen_ref_5_v08
- gen_ref_6 => gen_ref_6_v08
- gen_narr => gen_narr_v08

camt.054 IN

mx_camt054n_trx_sic_in_css

- msg#do_ovr_hdl_msg => msg#do_ovr_hdl_msg_v08
- msg#ovr_hdl_msg => msg#ovr_hdl_msg_v08

camt.056 IN

mx_camt056n_trx_in_css

- msg#order_type_id => msg#order_type_id_v08
- msg#paytb_bp_id = msg#paytb_bp_id_v08
- doc#fill_extn => doc#fill_extn_v08
- doc#valid => doc#valid_v08

camt.056 OUT

mx_camt056n_trx_out_css

- inpay#ovr_assgnr => inpay#ovr_assgnr_v08
- inpay#ovr_assgne => inpay#ovr_assgne_v08

PAIN.001 CH

- All CSS unchanged (same structure is used for both)

5.3 Upgrading SIC for November 2022

To keep up to date with the SIC 2022 release upgrade, complete the following steps.

Avaloq has implemented some changes to the kernel code. The following changes *must* be made to keep customization in line with these changes.

The `netw_release_id` item of the `avq.msg.swift` base parameter defines the currently valid release for SWIFT, SIC, SEPA and PostFinance. This item must be set at the date of a release upgrade. If this base parameter is not set, or is set with the value from a previous release, the implementation assumes the lowest possible or previous release for a given stream.

The `netw_release_id` base parameter item of the `avq.msg.sic` base parameter is no longer used and does not need to be updated. Instead, the `netw_release_id` item of the `avq.msg.swift` base parameter is used for all S4 and PostFinance releases.

The changes for the 2022 release can be installed *before* the actual release date.

Message structures are extended to support the old and new releases. Outgoing message handlers are capable of completing the messages in both the old *and* new formats for each network.

The decision of which version to use is specified in the `netw_release_id` item of the `avq.msg.swift` base parameter.

5.3.1 Prerequisite

Before testing the changes delivered for SIC Standards Release 4.9 (November 2022), you must set the `netw_release_id` item of the `avq.msg.swift` base parameter to "swift$sss22" on the testing environment. When moving the change to production, you must also ensure that this value is set.

Once `netw_release_id` is set to "swift$sss22", all outgoing messages will use the ISO v2019 (that is, 2022 specification) SIC version structure. The meta_msg_id is unchanged (so settlement plans can also remain unchanged) but the associated structure (e.g. meta_msg_struct_id) will be the new version (and so will the associated message DDIC). For incoming, ACP will recognise old and new versions by namespace, and will choose the message structure appropriate to the version incoming. Therefore, the same MSG IN source will still be called.

The switch from unversioned CSS to versioned CSS is triggered by the setting of `netw_release_id = swift$sss22` or higher (and the resultant switch of message structure from old to new version). The change of message structure version means the message DDIC structure is different, but the `meta_msg_id` is unchanged (and so also no need to change settle plans). To test the structure in code, use the `meta_msg_struct_id`. For a full list of equivalent CSS options, see "Versioned CSS" on page 25.

If re-using code between old and new CSS versions in a procedure/function, the message parameter can only have one type. It is preferred to use the new version, but there is no static checking this is correct for the old version, and any path that is different will result in a runtime `tree#` exception.

5.3.2 Steps to Follow

1. On the productive Avaloq Core, on the date the release upgrade is productive on the SIC network, set the `netw_release_id` item of the `avq.msg.swift` base parameter to "swift$sss22".
2. Copy CSS from old version to new version (and adapt as required) and resolve any data dictionary conflicts.

   Any logic present in `ovr_msg` in CSS sources **must** to be ported and adapted in the `ovr_msg_v08` function. After the base parameter switch, `ovr_msg` will no longer be called.

3. Ensure that any underlying procedures or functions are compatible with both the old and new versions.
4. To assist with & handle path differences, there are message version agnostic functions/procedures in MX_UTIL. If not using these, a cast “with mem_msg_mx_pacs008n_v08/mem_msg_mx_pacs008n(i_msg) as m do” around any fields that are different between versions can be used to run correctly without runtime errrors.

   You can test the version difference either by using meta_msg_struct_id on the message, or by using msg_util.has_child_node.

   Example of using a MX_UTIL utility function for version differences in BIC/BICFI:

   ```
   mx_util.fin_instn#bic(i_msg.document.fi_to_fi_cstmr_cdt_trf.cdt_trf_tx_inf_list(1).cdtr_agt.fin_instn_id)
   ```

5. Ensure any rules dependent on `meta_msg_struct_id` are also added for the new structure ID for all message types. If using `meta_msg_id` instead, these can remain unchanged and will work for both versions.

5.3.3 Result

The SIC[4] implementation is up-to-date.

6 SECOM Customization Upgrade November 2022

6.1 SECOM Changes for November 2022

No kernel changes are delivered for the SECOM network. Changes from SWIFT MT are also inherited in SECOM messages, so to keep your system current for the SECOM November 2022 update, review the SWIFT MT changes and modify the relevant customization sources if required.

7 SEPA Customization Upgrade November 2021

To keep your system current, review the official specification and modify the relevant customization sources if required.

For more information, see *SEPA Release 2022 - Business User Guide (doc. ID: 1422)*.

7.1 DFÜ Changes for November 2021

The changes described below have been introduced based on the German Banking Industry Committee (Deutsche Kreditwirtschaft, DK) specification for the SEPA data formats for the customer-to-bank interface, which are, in turn, based on the EPC Implementation Guidelines.

7.1.1 Customer statement message according to ISO standard 20022

With effect from November 2021, to synchronize with the ISO20022 versions used in payments, the German banking industry will adapt the rules used in Customer Statement Messages to the ISO version V08 (2019 version) in November 2021.

This involves the following changes to camt.052/053/054:

- `BxTxCd/Cd` and `BkTxCd/Domn` are now filled in kernel (if `BkTxCd/Prtry` is not filled already e.g. default '+++').
- Any overwritten CAMT reports, or creation of new CAMT messages in CSS, should now use the V08 version.

Change `'with new mem_msg_mx_camt.052n'` to `'with new mem_msg_mx_camt052n*_v08*'`

- Any use of fields that have changed from V02 to V08 must be wrapped in a `with` statement for v08 to utilize accurate static checks in compilation.

If accessing Ntry/ValDate, Ntry/BookgDt and Ntry/Sts:

```
with mem_msg_mx_camt053n_v08
(i_msg).document.bk_to_cstmr_stmt.stmt_list(1) as stmt_v08 do
with stmt_v08.ntry_list(i_ntry_id) as ntry_v08 do
ntry_v08.val_dt.dt.val_date := evt(i_evt_id).val_date;
...
end with;
end with;
```

As `ntry.val_dt.opt.dt.val_date := evt(i_evt_id).val_date;` will fail at runtime since the '`opt`' node no longer exists.

Required action: Review any existing customization relating to the camt.052/053/054 message types to ensure that it is compatible with the 2021 specification.

7.2 Upgrading SEPA for November 2022

The changes made for the SEPA 2022 release upgrade affect the processing of messages for the DFÜ clearer. If these changes apply to you, complete the following steps to keep up to date with the SEPA 2022 release upgrade:

Steps to Follow

1. On the productive Avaloq Core, on the date the release upgrade is productive on the SIC network, set the `netw_release_id` item of the `avq.msg.swift` base parameter to "swift$sss22".

7.2.1 Result

The SEPA implementation is up to date for 2022.