SmartVista   
TMA/TMT/TM9 Module

SmartVista Standard

April 15, 2015

Contents

[Introduction 4](#_Toc417034548)

[History 4](#_Toc417034549)

[Document purpose 4](#_Toc417034550)

[Functionality description 4](#_Toc417034551)

[Architecture overview 5](#_Toc417034552)

[Use-Case View 6](#_Toc417034553)

[Purpose of TMA/TMT module processing 6](#_Toc417034554)

[TMA processing 6](#_Toc417034555)

[TMT Processing 7](#_Toc417034556)

[DCC Calculation Processing 7](#_Toc417034557)

[GUI use case of Interchange fee calulation functionality 8](#_Toc417034558)

[Audit 8](#_Toc417034559)

[Incident Manager 8](#_Toc417034560)

[Manual transaction processing 8](#_Toc417034561)

[Logical View 9](#_Toc417034562)

[Scope of use 9](#_Toc417034563)

[Structure diagram 9](#_Toc417034564)

[Process View 10](#_Toc417034565)

[TMA Process Diagram 10](#_Toc417034566)

[TMA Validation rules 11](#_Toc417034567)

[SVXP mapping rules 11](#_Toc417034568)

[TMT Process Diagram 12](#_Toc417034569)

[Main process 12](#_Toc417034570)

[Check session process 13](#_Toc417034571)

[DCC Calculation Process Diagram 13](#_Toc417034572)

[GUI Process diagram 13](#_Toc417034573)

[Audit 13](#_Toc417034574)

[Incident Management 13](#_Toc417034575)

[Manual Transaction Processing 13](#_Toc417034576)

[Phisical model 13](#_Toc417034577)

[Interfaces 13](#_Toc417034578)

[Hermes 13](#_Toc417034579)

[JMS 13](#_Toc417034580)

[Reconciliation ws 13](#_Toc417034581)

[Configuration ws 13](#_Toc417034582)

Introduction

History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Author | Details |
| 0.1 | 04.15.2015 | Boris Varshavsky | First version of the document |
| 0.2 | 05.08.2015 | Boris Varshavsky | Second version of the document |

Document purpose

The purpose of the document is description of module TMA/TMT.

Dictionary

* SV – SmartVista processing
* SVT – SmartVista total for TMT sessions
* DCC - Dynamic Currency Conversion is an option which allows customer to choose the currency of transaction which will be posted on customer’s account.
* SVXP – XML format for exchange with SV
* TMA – table which contains online authorizations
* TMT – table which contains session info and totals for session.
* SVNG – SmartVista New Generation

Functionality description

Basic requirements for TMA functionality:

* Check for new records in TMA
* Validate each TMA table row
* SVT calculation
* DCC Calculation
* Convert TMA table rows to SVXP structure
* Store clearing batch until SV2 request notification is send.
* Working with incidents (Reject management and manual transaction processing)
* Sending alerts to Hermes
* Receive diff from reconciliation and creating reversals for these transactions

Basic requirements for TMT functionality:

* Closing sessions
* Monitoring active sessions and sending alerts if sessions are hang.
* Send Totals to Reconciliation module
* Send close session message to SV2

Architecture overview



Use-Case View

Purpose of TMA/TMT module processing



TMA processing

**Actors:**

TMA table, replicated from Stratus - data provider

**Pre-conditions:**

Stratus performs replication of ISO8583 transactions to TMA table

**Post-conditions:**

SVXP batch file stored in BATCH\_DATA table

**Event flow:**

1. TMA process checks for new records in TMA table by scheduler.
2. If table contains new records, they are sent for processing one by one.
3. Module checks message types and function codes of record. If transaction doesn’t has financial impact, then sets status “SKIPPED” and further processing stops.
4. Module checks record for necessity of awaiting response. If record need to await response, then sets status “PENDED”. Record will be processed by TM9 Processing.
5. Module calculates session id and checks record session id.
6. If session has not been registered in TMA module, then sends OpenSession request to Reconciliation module.
7. For each record performs validations and total calculation.
8. If validation was unsuccessful, sets record status to error status, further processing stops.
9. If transaction is DCC performs DCC calculation.
10. Maps record into SVXP format. Validate SVXP message by XSD.
11. If validation was successful, adds record amounts to session total amounts.
12. When SVXP operation batch reaches batch size, store batch file.
13. When module receives request from SV2, starts uploading all stored batches into queue.
14. When module receives request from Reconciliation, then find all records by session\_id received from Reconciliation and send it in queue.
15. Use-case ends.

TMT Processing

**Actors:**

TMT table, replicated from Stratus, data provider

**Pre-conditions:**

Stratus performs replication of transactions to TMT table

**Post-conditions:**

Info about totals is sent to Reconciliation module

**Event flow:**

1. TMT process checks new records in TMT table by scheduler
2. If table contains new records, then module starts close session processing.
3. If field CMTMSEAD <> 1 in TMT table, that means, that module need to match TMT and TMR record for obtaining POS totals.
4. If message type = 1520, then saves total value into te\_total field in session table.
5. If message type = 1530, then saves total value into tr\_total field in session table.
6. Module sends CloseSession message with totals to Reconciliation module by web-service and stores batch file with end session parameters for sending to SV2.
7. Sets session status to “CLOSED”
8. Module should monitor sessions while they are not closed. If a session is not closed for a configured period of time, module should generate alert to Hermes.
9. Module can access transactions after session is closed. In this case recalculates session total, sets session status to “ACTIVE”, and starts end of session with sending alert to Hermes.
10. Use-case ends.

TM9 Processing

**Actors:**

TM9 table, replicated from Stratus, data provider

**Pre-conditions:**

Stratus performs replication of records to TM9 table

**Post-conditions:**

Records are joined with TMA records and processed

**Event flow:**

1. TM9 process checks new records in TM9 table by scheduler
2. If table contains new records, then module starts record processing.
3. Join TM9 and TMA record. If TMA record was successfully joined, then sets status “PROCESSING”
4. TM9+TMA record must be processed as usual TMA record in TMA processing part, started from paragraph 5.
5. Use-case ends.

GUI use case of TMA/TMT module functionality



Audit

Incident Manager

Manual transaction processing

**Actors:**

GUI Form “Incident manager”

SV2, Reconciliation module – data consumer

**Pre-conditions:**

Record with status “ERROR” has been received

**Post-conditions:**

Record sent for reconciliation

**Event flow:**

1. User checks records with status “ERROR”
2. User selects necessary record and click button “Edit and resend”
3. “Edit record” form is opened.
4. User edits current record and click button “Resend”
5. Flag “MANUALLY\_RESEND” stored in database
6. TMA process starts.

SV2, Reconciliation module – data consumer

**Pre-conditions:**

New record has been registered in GUI

**Post-conditions:**

Record sent for reconciliation

**Event flow:**

1. User registers new transaction in UI.
2. “Create record” form is opened.
3. User create new record and click button “Send”
4. Flag “MANUALLY\_RESEND” stored in database
5. TMA process starts.

SV2, Reconciliation module – data consumer

**Pre-conditions:**

New total has been registered in GUI

**Post-conditions:**

New total value sent for processing

**Event flow:**

1. User registers new total in UI.
2. “Create total” form is opened.
3. User create total and click button “Send”
4. TMT process starts.

SV2, Reconciliation module – data consumer

**Pre-conditions:**

User clicks button re-close session

**Post-conditions:**

TMT process is started again

**Event flow:**

1. User clicks button “re-close session” in UI.
2. TMT process starts.

Integration with SV2



Load data to SV2

**Actors:**

TMA module – producer

SV2 – consumer

**Pre-conditions:**

TMA/TMT module receives “LoadData” request from SV2

**Post-conditions:**

SVXP batches sent to the SV2

**Event flow:**

1. SV2 send “LoadData” request
2. TMA module return count of all stored batches from BATCH\_DATA table with status “READY\_TO\_SEND”
3. TMA module select all stored batches from BATCH\_DATA table with status “READY\_TO\_SEND”
4. TMA module packs into SVXP envelope wrapper format (format described in BPC\_SVNG\_TARGET\_ARCH\_ENG) and send all selected batches to queue received from “LoadData”
5. TMA module sends “LoadDone” notification message to SV2
6. After finishing upload SV2 send to TMA module “EndLoad” request with count loaded records and batches. If count of loaded records or count of loaded batches is equal to count of sended records and batches, then sets status “SEND” to all batch records, else status doesn’t change.

Integration with Reconciliation



Load data to reconciliation

**Actors:**

TMA module – producer

Reconciliation – consumer

**Pre-conditions:**

TMA/TMT module receives REST request from reconciliation with session id.

**Post-conditions:**

SVXP batch with transactions for concrete session id sent to the reconciliation

**Event flow:**

1. Reconciliation send REST request with session id.
2. TMA module, select all transactions for this session id and maps it to SVXP batches.
3. TMA module send all to reconciliation queue

Receiving diff from reconciliation

**Actors:**

TMA module – consumer

Reconciliation module – producer

**Pre-conditions:**

TMA/TMT module receives SVXP batches with diff from reconciliation

**Post-conditions:**

TMA/TMT module creates reversals and processed them.

**Event flow:**

1. TMA/TMT module receives SVXP batches with diff from reconciliation.
2. TMA module, maps SVXP messages with record in TMA table.
3. TMA module creates reversal record in TMA from mapped record.
4. Reversal records processing by TMA module.

Process View

TMA Process Diagram



**Description of steps:**

1. **Check for new records in TMA table**

Module checks for new records by scheduler.

Input parameters: cron expression(from configuration table).

1. **Sending each record for processing**

* For each new record sets status “PREPARED” and sends record for processing.
* Checks message type and function code of the record

|  |  |  |
| --- | --- | --- |
| Message type (CMALMENS) | Function code (CM4LFMEN) | Description |
| 1100 | 100,101,103,105,181,185,187 |  |
| 1120 | 100,101,103,105,181,185,187 |  |
| 1121,1122,1123 | All codes |  |
| 1421,1422,1423 | All codes |  |

For records with message types and function codes described above sets status “SKIPPED” (it’s pre-authorization and they doesn’t has financial impact), for all other records sets status “PROCESSING”.

* If message, according to rules, has to await response record in TM9 table, then sets status “PENDED”. Processing stops, until response record in TM9 table is appeared.

Rules, according to which record sets to status “PENDED”

1. C0DLCIE3 or C0DLCIE3\_T01 field starts with ‘82’
2. Field CMAMDAT\_ONL <> 1 or field substring of field CMAVDAT\_ONL(17:1) <> 1 or field CMAVESPR in (4,5,6).

* Module checks record countability: if field CMAQSEAD != null, then transaction is “acquirer-countable”, if field CMAQSERE != null, then transaction is “issuer-countable”.
* If record countable, then module checks record session id (Session id generation rules are described below).

1. **Session ID generation rules.**

|  |  |  |  |
| --- | --- | --- | --- |
| Application ID | Type | Fields | Format |
| 01 | Acquirer | C0DLCIER (first 2 symbols) -  C0DLCIER (last 4 symbols) \_  CMAQSEAD | 01\_XXXXXX-XXXXXX |
| 01 | Issuer | C0DLCIE3 (first 2 symbols) -  C0DLCIE3 (last 4 symbols) \_  CMAQSERE | 01\_XXXXXX-XXXXXX |
| 02 | Acquirer | C0DLCIER (first 2 symbols) -  C0DLCIER (last 4 symbols) \_  CMAQSEAD | 02\_XXXXXX-XXXXXX |
| 02 | Issuer | C0DLCIE3 (first 2 symbols) -  C0DLCIE3 (last 4 symbols) \_  CMAQSERE | 02\_XXXXXX-XXXXXX |
| 07 | Acquirer | C0DLCIER (first 2 symbols) -  CMALCSBM+  c43leca +  c43lcome +  c43cchec -  cmacterm \_  CMAQSEAD | 07\_XXXXXX -XXXXXX |
| 13 | Acquirer | C0DLCIER (first 2 symbols) -  CMALCSBM \_  CMAQSEAD | 13\_XXXXXX-XXXXXX |
| 23 | Acquirer | C0DLCIER (first 2 symbols)\_  CMALCSBM+  c43leca +  c43lcome +  c43cchec -  cmacterm -  CMAQSEAD | 23\_XXXXXX -XXXXXX |
| 24 | Acquirer | C0DLCIER (first 2 symbols)\_  Cmalcsbc-  CMAQSEAD | 24\_XXXXXX-XXXXXX |
| 24 | Issuer | C0DLCIE3 (first 2 symbols)\_  C0DLCIE3 (last 4 symbols) -  CMAQSERE | 24\_XXXXXX-XXXXXX |
| 42 | Acquirer | C0DLCIER (first 2 symbols)\_  Cmaccaje-  CMAQSEAD | 42\_XXXXXX-XXXXXX |
| 43 | Acquirer | C0DLCIER (first 2 symbols)\_  C0DLCIER (last 4 symbols)-  CMAQSEAD | 43\_XXXXXX-XXXXXX |
| 47 | Acquirer | C0DLCIER (first 2 symbols)\_  CMALCSBM+  c43leca +  c43lcome +  c43cchec -  cmacterm-  CMAQSEAD | 47\_XXXXXX-XXXXXX |
| 63 | Acquirer | C0DLCIER (first 2 symbols)\_  CMALCSBM+  c43leca +  c43lcome +  c43cchec -  cmacterm-  CMAQSEAD | 63\_XXXXXX- XXXXXX |
| 67 | Acquirer | C0DLCIER (first 2 symbols)\_  CMALCSBM+  c43leca +  c43lcome +  c43cchec -  cmacterm-  CMAQSEAD | 67\_XXXXXX- XXXXXX |
| 79 | Acquirer | C0DLCIER (first 2 symbols)\_  CMALCSBM+  c43leca +  c43lcome +  c43cchec -  cmacterm-  CMAQSEAD | 79\_XXXXXX- XXXXXX |

For all others applications, session id calculated like in application id = 1.

* If session id doesn’t exist in session table, creates new session with status “ACTIVE”, and sends “OpenSession” message to Reconciliation.
* If session id exists and status “CLOSED”, then module sends alert to Hermes and continues processing.
* If session id exists, status “CLOSED” and close date greater than current date + configurable parameter. Then further processing stops, sets status “ERROR” and module sends alert to Hermes.
* If session id exists and status “ACTIVE”, then continue processing.

1. **Record validation**

* Builds validation list order by validation priority table.
* Record passes through list of validations one by one
* If validation failed, and validation type is “CRITICAL”, sets status “ERROR” for this record, stores error code, comment and validator id. Further processing stops.
* If validation failed, and validation type is “NON\_CRITICAL”, sets status “ERROR” for this record, stores error code, comment and validator id. Further processing continues, but at the end of validation list further processing is stopped.

1. **SVXP mapping**

* Message maps into SVXP format, mapping rules described in annex TMA.xls and Mapping.xls.
* SVXP “Operation” tag, which contains result data validates by XSD. If validation fails, then record sets status “ERROR”, further processing stops.

1. **DCC Calculation**

* If function code (CM4LFMEN) equals 184(DCC transaction), then module performs dcc calculation. Algorithm of DCC calculation described below.

1. **SVT Calculation**

If validation was successful, module performs SVT calculation. Module updates appropriate total field in TMT\_session table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Source (PRICE)** | **TMT\_SESSION field** | **Condition** |
| Number of credits | S74 | CMTIABON | Add 1 if CMALMENS field = 1120, and concatenation of fields CM2LTRSP, CM3LTCTA,CM3QNULO between "200000" and "290000" inclusive |
| Number of credits from Cancellations | S75 | CMTIANAB | Add 1 if CMALMENS field = 1420 and concatenation of fields CM2LTRSP, CM3LTCTA,CM3QNULO between "000000" and "190000" inclusive |
| Number of debits | S76 | CMTICARG | Add 1 if CMALMENS field = 1100,1120, and concatenation of fields CM2LTRSP, CM3LTCTA,CM3QNULO between "000000" and "190000" inclusive |
| Number of debits from Cancellations | S77 | CMTIANCA | Add 1 if CMALMENS field = 1420 and concatenation of fields CM2LTRSP, CM3LTCTA,CM3QNULO between "200000" and "290000" inclusive |
| Number of Service Transactions | S80 | - | Contains zero value |
| Number of Authorizations | S81 | - | Contains zero value |
| Number of collections / payments | S85 | - | Contains zero value |
| Amount of Credits | S86 | CMTJABON | Add CMAJIMCO (or CMAJIMAD is null) field value if CMALMENS field = 1120, and concatenation of fields CM2LTRSP, CM3LTCTA,CM3QNULO between "200000" and "290000" inclusive |
| Amount of Credits from Cancellations | S87 | CMTJANAB | Add CMAJIMCO (or CMAJIMAD is null) field value if CMALMENS field = 1420and concatenation of fields CM2LTRSP, CM3LTCTA,CM3QNULO between "000000" and "190000" inclusive |
| Amount of Debits | S88 | CMTJCARG | Add CMAJIMCO (or CMAJIMAD is null) field value if CMALMENS field = 1100,1120, and concatenation of fields CM2LTRSP, CM3LTCTA,CM3QNULO between "000000" and "190000" inclusive |
| Amount of Debits from Cancellations | S89 | CMTJANCA | Add CMAJIMCO (or CMAJIMAD is null) field value if CMALMENS field = 1420and concatenation of fields CM2LTRSP, CM3LTCTA,CM3QNULO between "200000" and "290000" inclusive |
| Net position | S97 | CMTJTINT | It will be obtained applying the formula:  S97 = S86 + S87 - S88 - S89+ S109 - S110 |
| Amount of Credits for Charge-backs | S105 | - | Contains zero value |
| Amount of Debits for Charge-backs | S106 | - | Contains zero value |
| Number of Credits for Charge-backs | S107 | - | Contains zero value |
| Number of Debits for Charge-backs | S108 | - | Contains zero value |
| Amount of Installment Credit | S109 | CMTJTABO | Add CMAJCUO1, CMAJCUO2, CMAJCUO3, CMAJCUO4 value if they are positive |
| Amount of Installment Debit | S110 | CMTJTCAR | Add CMAJCUO1, CMAJCUO2, CMAJCUO3, CMAJCUO4 value if they are negative |

1. **Add record to batch**

If batch size reaches batch size set in configuration table, store it in BATCH\_DATA table with status “READY\_TO\_SEND” and sets status “PROCESSED” for all messages in batch module.

TMA Validation rules

P18 Field Validation

P18 field in PRICE specification is equal to “C41LACTI” column in TMA\_TMT table. Module validates column value by dictionary (list of values in PRICE specification p. 6 p.12)

P24 Field Validation

P24 field in PRICE specification is equal to “CM4LFMEN” column in TMA\_TMT table. Module validates column value by dictionary (list of values in PRICE specification p. 6 p.55)

P25 Field Validation

P25 field in PRICE specification is equal to “CM5LRAZM” column in TMA\_TMT table. Module validates column value by dictionary (list of values in PRICE specification p.6 p.62)

P39 Field Validation

P39 field in PRICE specification is equal to “CM1LRESP” column in TMA\_TMT table. Module validates column value by dictionary (list of values in PRICE specification p.6 p.146)

P40 Field Validation

P40 field in PRICE specification is equal to “CA5LSRTJ” column in TMA\_TMT table. Module validates column value by dictionary (list of values in PRICE specification p.6 p.155)

P50 Field Validation

P50 field in PRICE specification is equal to “CE0LMON3” column in TMA\_TMT table. Module validates column value. It should be “978” (PRICE specification p.3 p.96)

P57 Field Validation

P57 field in PRICE specification is equal to “CMAQPPAU” column in TMA\_TMT table. Module validates column value (rules in PRICE specification p. 3 p.171)

Sequence diagram



TMT Process Diagram

Main process



Description of steps:

1. **Check new records in TMT table**

Module checks TMT table for new records with message types (column ’CMTLMENS‘) ’1520‘ or ’1530‘. For other message types: stores status “SKIPPED”, further processing stops.

Input parameters: cron expression(from configuration table).

1. If field CMTMSEAD <> 1 in TMT table, that means, that module need to match TMT and TMR record for obtaining POS totals. Join rules are represented above.

|  |  |  |
| --- | --- | --- |
|  | TMT | TMR |
| P11 | CMTQIDTR | CM9QIDTR |
| P12 | CMTPTRAN | CM9PTRAN |
| P32 TRANSFORM | C0DLCIER | C0DLCIER |

If TMT and TMR records couldn’t join, then stores status “error”, further processing stops.

1. **Calculate Session ID**

|  |  |
| --- | --- |
| Application ID(CP0LAPPL) | Rule |
| 01 | C0DLCIER(first 2 characters)\_C0DLCIER(next 4 characters)-CMTQSEAD |
| 02 | C0DLCIER(first 2 characters)\_C0DLCIER(next 4 characters)-CMTQSEAD |
| 07 | c0dlcier(first 2 characters)\_c00lcsb+c43leca+c43lcome+c43cchec-cmtqncta-CMTQSEAD |
| 13 | C0DLCIER(first 2 characters)\_C0DLCIER(next 4 characters)-CMTQSEAD |
| 22 | C0DLCIER(first 2 characters)\_C0DLCIER(next 4 characters)-CMTQSEAD |
| 23 | c0dlcier(first 2 characters)\_c00lcsb+c43leca+c43lcome+c43cchec-cmtqncta-CMTQSEAD |
| 42 | C0DLCIER(first 2 characters)\_ C20LORDECMTQSEAD |
| 43 | C0DLCIER(first 2 characters)\_C0DLCIER(next 4 characters)-CMTQSEAD |
| 47 | c0dlcier(first 2 characters)\_c00lcsb+c43leca+c43lcome+c43cchec+cmtqncta-CMTQSEAD |
| 63 | c0dlcier(first 2 characters)\_c00lcsb+c43leca+c43lcome+c43cchec+cmtqncta-CMTQSEAD |
| 67 | c0dlcier(first 2 characters)\_c00lcsb+c43leca+c43lcome+c43cchec+cmtqncta-CMTQSEAD |
| 79 | c0dlcier(first 2 characters)\_c00lcsb+c43leca+c43lcome+c43cchec+cmtqncta-CMTQSEAD |

1. **Check message type of record**

Module creates record with new total values in TMT\_SESSION\_MESSAGE table with appropriate type. If session has already closed, then sends alert to Hermes, further processing stops.

If TMT\_SESSION\_MESSAGE contains ‘1520’ and ‘1530’ (‘TE’ and ‘TR’) total values, all total values was calculated. Set status “READY\_TO\_SEND”.

1. **Send message to reconciliation**

Module sends “CloseSession” message with totals to Reconciliation module.

1. **Send message to SV2**

Module creates SVXP message with net total value (S97). Total value sends in operation tag with Oper\_type = OPTP0250. And store it in BATCH\_DATA table.

|  |  |  |
| --- | --- | --- |
| Field | SVXP tag |  |
| TMT\_SESSION table column S97 | Operation/oper\_amount/amount\_value |  |
| STTT0000 constant | Sttl\_type |  |
| OPTP0250 | Oper\_type |  |
| TMT\_SESSION table sessionId column | Operation/auth\_data/auth\_tag/tag\_id = 5001  Operation/auth\_data/auth\_tag/tag\_value |  |
| Net total amount for Intra-Servired | Operation/auth\_data/auth\_tag/tag\_id = 5013  Operation/auth\_data/auth\_tag/tag\_value |  |
| Net total amount for National Internetworks | Operation/auth\_data/auth\_tag/tag\_id = 5014  Operation/auth\_data/auth\_tag/tag\_value |  |
| Net total amount for VISA | Operation/auth\_data/auth\_tag/tag\_id = 5015  Operation/auth\_data/auth\_tag/tag\_value |  |
| Net total amount for CUP | Operation/auth\_data/auth\_tag/tag\_id = 5016  Operation/auth\_data/auth\_tag/tag\_value |  |
| Net total amount for MasterCard | Operation/auth\_data/auth\_tag/tag\_id = 5017  Operation/auth\_data/auth\_tag/tag\_value |  |

Example of total request:

<tns:clearing xmlns:tns="http://bpc.ru/sv/SVXP/clearing">

<tns:file\_type>12345678</tns:file\_type>

<tns:operation>

<tns:oper\_type>OPTP0250</tns:oper\_type>

<tns:sttl\_type>STTT0000</tns:sttl\_type>

<tns:oper\_amount>

<tns:amount\_value>S97 value</tns:amount\_value>

<tns:currency>EUR</tns:currency>

</tns:oper\_amount>

<tns:auth\_data>

<tns:auth\_tag>

<tns:tag\_id>5001</tns:tag\_id>

<tns:tag\_value>sessionId value</tns:tag\_value>

</tns:auth\_tag>

<tns:auth\_tag>

<tns:tag\_id>5013</tns:tag\_id>

<tns:tag\_value>Net total amount for Intra-Servired</tns:tag\_value>

</tns:auth\_tag>

<tns:auth\_tag>

<tns:tag\_id>5014</tns:tag\_id>

<tns:tag\_value>Net total amount for National Internetworks</tns:tag\_value>

</tns:auth\_tag>

<tns:auth\_tag>

<tns:tag\_id>5015</tns:tag\_id>

<tns:tag\_value>Net total amount for VISA</tns:tag\_value>

</tns:auth\_tag>

<tns:auth\_tag>

<tns:tag\_id>5016</tns:tag\_id>

<tns:tag\_value>Net total amount for CUP</tns:tag\_value>

</tns:auth\_tag>

<tns:auth\_tag>

<tns:tag\_id>5017</tns:tag\_id>

<tns:tag\_value>Net total amount for MasterCard</tns:tag\_value>

</tns:auth\_tag>

</tns:auth\_data>

</tns:operation>

</tns:clearing>

Additional processes

Check session process



**Exceptions**

1. Step 5 - Send to Reconciliation error

Set status to ‘ERROR’. Send alert to Hermes.

1. Step 6 - Send to SV error

Set status to ‘ERROR’. Send alert to Hermes.

1. Step 7 - Check session error

Send alert to Hermes.

TE total receiving.

If TE total has not been received for configurable period of time until TR total was received, then TR total value copies to TE total value.

Sequence diagram



DCC Calculation Process Diagram

1. **Check for DCC records**

If message has function code (field CM4LFMEN) = 184, 284, then calculates DCC fee value.

1. **Request exchange rates and fees**

Exchange rates and fees are received from configuration module by queue.

1. **DCC Calculation**

Module performs calculation only for P24 field (CM4LFMEN) with DCC type(184, 284).

If P03 field (CM2LTRSP) eq “1” it’s ATM transaction, if P03 field eq “0” it’s POS transaction.

Below is the formula for DCC amount calculation:



is transaction amount including ‘mark-up’ amount in transaction currency.

is transaction amount in transaction currency. (CMAJIMAD)

is ‘mark-up’ rate. It gets from configuration module. Configuration module might specify mark-up rate for concrete merchant in POS transactions. If module doesn’t have mark-up rate for entity and merchant, then module uses mark-up rate for entity only.

is mark-up value.

Convert this amount ( calculated in the previous point) from transaction currency to the cardholder billing currency using SmartVista exchange rates (from table TEJ).

. Currency exchange rate gets from CMAICAMC column.

If parameter ‘DCC availability’ in settings from configuration module is set as ‘false’ SmartVista will generate alert.

If the mark-up fee value is empty system shall check DCC parameters applying for whole Entity as a default value (without specifying merchant Id).

1. **Compare dcc value**

DCC module compare calculated value with value calculated by Stratus (if field CMALTCU1 = 74, then CMAJCUO1 contains DCC value, similarly for CMALTCU2, CMALTCU3, CMALTCU4). In case of discrepancies DCC module will generate alert to Hermes. Discrepancies must be available in web-interface and XLS file.

1. **DCC unload**

DCC values may be unloaded into XLS format from web-interface. Unload full TMA record with calculated DCC value.

TM9 Process Diagram

Main process



Description of steps

1. Module checks TM9 table for new records
2. If TM9 table contains new record, then join TM9 record with TMA record in status “PENDED”. If record in status “PENDED” not found, then sends alert to Hermes.

Join rules for TM9-TMA records:

Join key for tables is (P-32) + (P-12) + (P-11) (according PRICE specification).

* For TMA
* If field CMAMCONX == 1, then CMALCSBM field is used as P-32, otherwise C0DLCIER
* P-12 = CMAPTRAN
* P-11 = CMAQIDTB
* For TM9
* P-32 = C0DLCIER
* P-12 = CM9PTRAN
* P-11 = CM9QIDTR

1. Sets status in TMA table to “PROCESSING” and record processed as usual TMA record starting from paragraph 5 in TMA processing.

Process diagram

Audit

Incident Management

Manual total execution

User must have opportunity to add TE or TR total from interface of reject manager.

Manual Transaction Processing

Phisical model



Interfaces

Hermes

Server WS

|  |  |  |
| --- | --- | --- |
| Service | Method | Description |
| TmaTmtServerService | GetTransactions | Request list of transaction by filter parameters |
|  | GetSessions | Request session list by filter parameters |
|  | EditTransaction | Edit current transaction |
| TmaTmtTotalServerService | GetUpdatedTotals | Request current totals values(for non-closed session) |

Client ws

|  |  |  |
| --- | --- | --- |
| Service | Method | Description |
| ReconAPI | closeSession | Close session and send totals. |
| ReconAPI | openSession | Open session. |

Queues

Message Queues

|  |  |
| --- | --- |
| Queue | Description |
| TmaTmtQueue | Queue in which module will send SVXP batch |
| ConfigurationQueue | Queue for cooperation with Configuration module. |