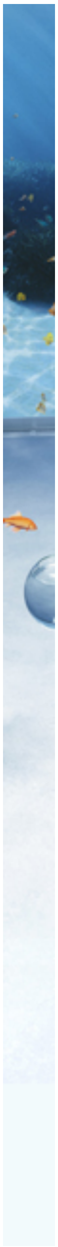

Mx G2000 to Mx.3 Migration Data model and Reporting Impact Analysis

MX.3

2008/08/13

MUREX DEMO

Murex Integration DMM Team

Table of Contents

Table of Contents

Mx G2000 to Mx.3 Migration Data model and Reporting Impact Analysis

Table of Contents

Mx G2000 to Mx.3 Migration Data model and Reporting Impact Analysis

Table of Contents

Mx G2000 to Mx.3 Migration Data model and Reporting Impact Analysis

This document details the analysis on the MX data model changes and their impact on Reporting for a Mx G2000 to Mx.3 migration.

For each topic, the data model differences are detailed and a mapping is proposed for the implementation.

This analysis is useful for a better understanding of the Mx.3 data model and for the migration of:

- Filters formulas, horizontal fields formulas, Mreports, unions and all other tools accessing directly the MX data model
- Dynamic tables

1 – Organization

1.1 – Data model schema

1.1.1 – In Mx G2000

Figure 4 represents the data model in Mx G2000.

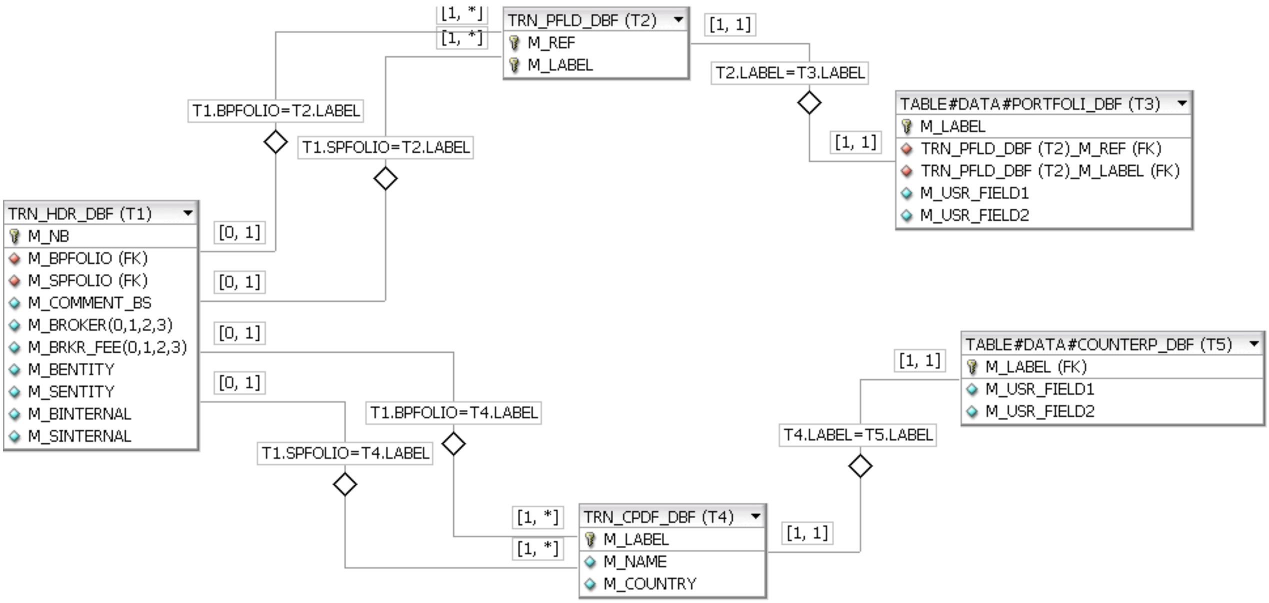

Figure 1: Link between TRN_HDR_DBF and organization tables (2.10)

1.1.2 – In Mx.3

Figure 5 represents the data model in Mx.3.

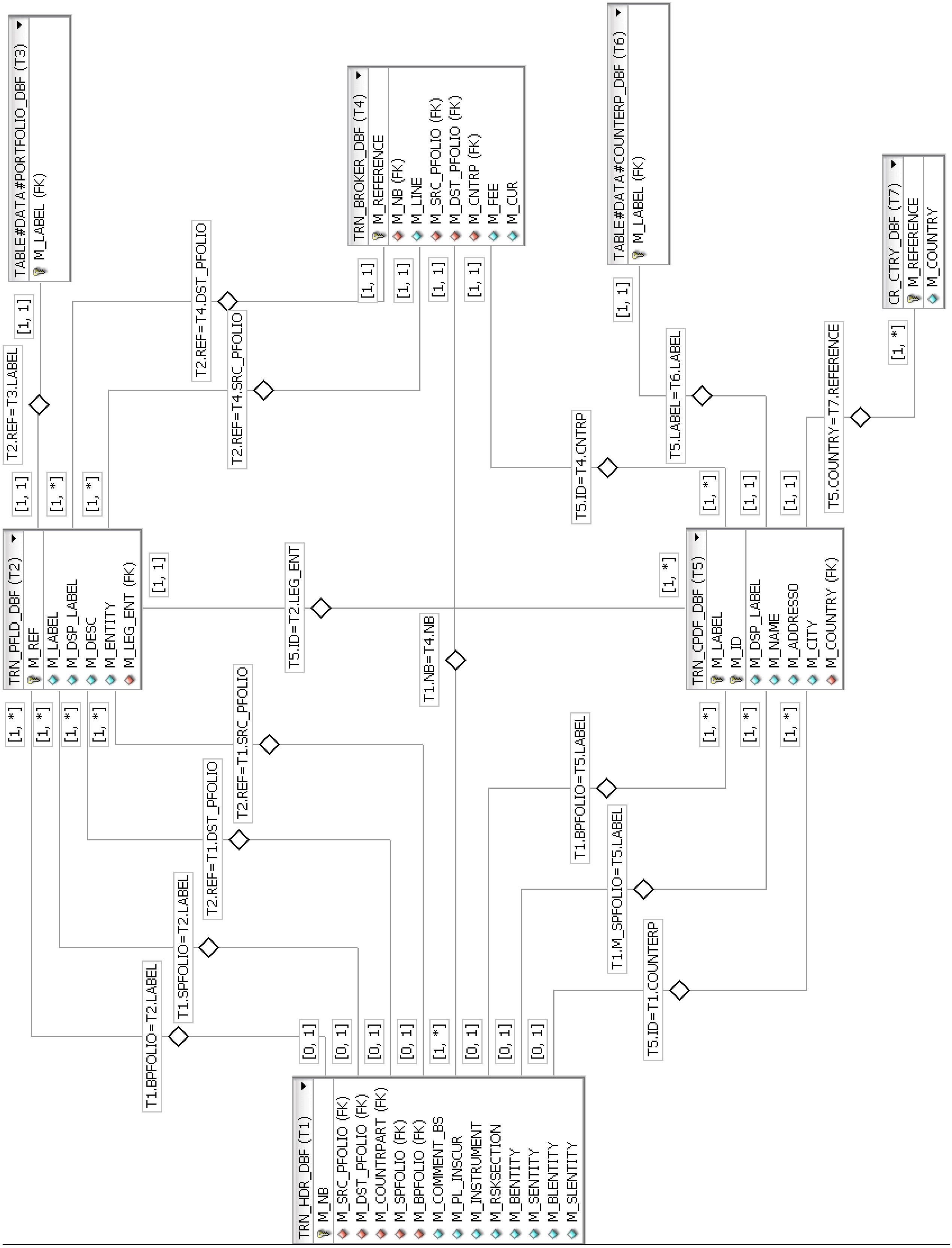

Figure 2: Link between TRN_HDR_DBF and organization tables (Mx.3)

1.2 – Business parties and portfolios

1.2.1 – Business parties storage

In Mx G2000, business parties storage can be configurable from a Configurator session thanks to the flag "Activate unified counterpart".

If this flag is set to "No", business parties are stored in three different tables: counterparties in TRN_CPDF_DBF, brokers in BROKER#BRO_DEF_DBF and issuers in SE_ISS_DBF.

If this flag is set to "Yes", business parties are stored in two different tables as brokers will be stored in the counterparty table.

In Mx.3, this flag has been removed and all business parties (counterparties, brokers and clearers) are now stored in the counterpart table TRN_CPDF_DBF.

Sample to retrieve the list of counterparties in Mx.3:

```
select * from TRN_CPDF_DBF where (M_BANK='Y' or M_CLEARER='Y' or M_CLIENT='Y' or M_ISSUER='Y' or
M_STATE='Y' or M_CORPORATE='Y' or M_REFENTITY='Y' or M_GARANTOR='Y' or M_CUSTOMER='Y' or
M_FICTIVE='Y' or M_INT_PARTY='Y' or M_GROUP='Y'or M_SUBSIDIARY='Y' or M_BRANCH='Y' or M_OTHER='Y' or
(M_BROKER='N' and M_ENTITY='N'))
```

1.2.2 – Couterparties and portfolios

1.2.2.1 – In Mx G2000

In Mx G2000, information related to counterparties is stored in the table TRN_CPDF_DBF and the table TABLE#DATA#COUNTERP_DBF for counterparty user definable fields. Information related to portfolios is stored in the table TRN_PFLD_DBF and the table TABLE#DATA#PORTFOLI_DBF for portfolio user definable fields.

For counterparties as well as for portfolios, the field M_LABEL was used as a table key and also as a display field.

The following fields are also available: M_BPFOLIO, M_SPFOLIO, M_COMMENT_BS (buy/sell). The fields M_BPFOLIO and M_SPFOLIO contain portfolios or counterparties according to the buy/sell signification and whether the trade is internal or external. These fields can be used in a join to retrieve:

- the portfolio source (trades) : M_LABEL = M_BPFOLIO if (M_COMMENT_BS = "B") or M_SPFOLIO if (M_COMMENT_BS = "S")
- the portfolio destination (internal trades): M_LABEL = M_BPFOLIO if (M_BINTERNAL = "Y") or M_SPFOLIO if (M_SINTERNAL = "Y")
- the counterparty destination: M_LABEL = M_BPFOLIO if (M_BINTERNAL = "N") or M_SPFOLIO if (M_SINTERNAL = "N")

The country label was stored in the field M_COUNTRY of the table TRN_CPDF_DBF.

1.2.2.2 – In Mx.3

In Mx.3, the same tables keep the same information but have been enriched with new fields:

1.2.2.2.1 – Fields related to IDs and labels

M_DSP_LABEL contains the counterparty or portfolio display label. It should be used for display purposes only.

In Mx G2000, labels were also used as joining keys whereas in Mx.3 numerics are used instead. These columns, when inherited from Mx G2000, become hybrid: they contain the label for migrated records and the ID for newly added records. Therefore, when joining with other tables, one of the following fields should be used depending on the type of the current column:

- M_ID for counterparties or M_REF for portfolios contains the counterparty or portfolio ID. It should be used to join with columns of type numeric.
- M_LABEL used in Mx G2000 still exists for compatibility purposes. It should be used to join with columns of type character.

To link with the table TABLE#DATA#COUNTERP_DBF in Mx.3, it is necessary to use the hybrid field TRN_CPDF_DBF.M_LABEL.

Note that in Mx.3, since references are used to identify a row, new parser functions have been created to look up:

- the counterparty label by the display label: CTRP_LBL
- the counterparty by the displayed label: CTRP_REF
- the portfolio reference by the portfolio label PTF_REF

They are mostly useful to simplify pre−filter formulas in dynamic tables.

1.2.2.2.2 – Fields related to source and destination

The existing fields are maintained, but new fields are available notably to facilitate the retrieval of the source and the destination:

- M_SRC_PFOLIO: contains the source portfolio (always a portfolio)
- M_DST_PFOLIO: contains the destination portfolio or 0 if the destination is a counterparty (always a portfolio)
- M_COUNTERPART: contains the destination counterparty (always a counterparty)

Impact on Migration: The existing fields such as M_BPFOLIO and M_SPFOLIO can be used, but for new usage, new fields such as M_DST_PFOLIO should be used.

1.2.2.2.3 – Fields related to Country

The field M_COUNTRY in the table TRN_CPDF_DBF stores the reference of the country. A join with the table CR_CTRY_DBF is now necessary to display the country label.

In Mx.3, dynamic tables have been enriched with new fields to display the flow counterparties in addition to the trade counterparty. These are listed in section 1.2.3.2 Dynamic table impact.

1.2.3 – Fields Mapping

1.2.3.1 – Data model impact

V2.10 table	V2.10 field	Description	V3.1 table	V3.1 table	Description
TRN_CPDF_DBF	M_LABEL char (15)	counterparty label (label and key)	TRN_CPDF_DBF	M_LABEL char (15) if used as a join, or M_ID (numeric) if used as a join, or M_DSP_LABEL char (35) if used	M_LABEL is a hybrid column and is a key, M_ID is a key, M_DSP_LABEL is a label and not a key

				as a display	
TRN_CPDF_DBF	M_NAME	counterparty full name	TRN_CPDF_DBF	M_NAME	counterparty full name
None	None		TRN_CPDF_DBF	M_DSP_LABEL char (35)	Display label
None	None		TRN_CPDF_DBF	M_ID	Counterparty id
TRN_CPDF_DBF	M_COUNTRY		CR_CTRY_DBF	M_COUNTRY	Country label
TABLE#DATA#COUNTERP_DBF	M_LABEL		TABLE#DATA#COUNTERP_DBF	M_LABEL	
TRN_PFLD_DBF	M_LABEL		TRN_PFLD_DBF	M_LABEL	
TRN_PFLD_DBF	M_REF		TRN_PFLD_DBF	M_REF	
None	None		TRN_PFLD_DBF	M_DSP_LABEL	
TABLE#DATA#PORTFOLI_DBF	M_LABEL		TABLE#DATA#PORTFOLI_DBF	M_LABEL	

1.2.3.2 – Dynamic table impact

Dynamic table	V2.10 field	V3.1 table	Description
ALL	TP_CNTRP	• TP_CNTRPID for some joins • TP_CNTRPRF for other joins • TP_CNTRP for displaying label	• Trade Counterparty label/id (hybrid column) • Trade Counterparty reference (always id) • trade Counterparty display label (always label). Note that TP_CNTRP contains M_LABEL of TRN_PLD for internal deals
ALL	TP_CNTRPLB	TP_CNTRPFN	Counterparty full name
TRNRP_DT	None	DT_SPTF	Flow source portfolio
TRNRP_DT	None	DT_DPTF	Flow destination portfolio
TRNRP_DT	None	DT_CTP	Flow counterparty label (display label)
TRNRP_DT	None	DT_CTPID	Flow counterparty ID (label)
TRNRP_DT	None	DT_CTPRF	Flow counterparty ref
TRNRP_DT	None	DT_CTPFN	Flow counterparty full name
TRNRP_DT	None	DT_CTPFN	Flow counterparty full name
TRNRP_DT	None	DT_DESTLB	Flow destination label (counterparty or portfolio display label)
TRNRP_CS	None	F_SPTF	Flow source portfolio
TRNRP_CS	None	F_DPTF	Flow destination portfolio
TRNRP_CS	None	F_CTP	Flow counterparty label (display label)
TRNRP_CS	None	F_CTPID	Flow counterparty ID (label, duplicate existing field F_CTRP to rename it)
TRNRP_CS	None	F_CTPRF	Flow counterparty ref
TRNRP_CS	None	F_CTPFN	Flow counterparty full name

TRNRP_CS	None	F_DESTLB	Flow destination label (counterparty or portfolio display label)
TRNRP_MK	None	V_STLSPTF	Source portfolio of current settlement flow
TRNRP_MK	None	V_STLDPTF	Destination portfolio of current settlement flow
TRNRP_MK	None	V_STLCTP	Counterparty label of current settlement flow
TRNRP_MK	None	V_STLCTPID	Counterparty ID of current settlement flow
TRNRP_MK	None	V_STLCTPRF	Counterparty ref of current settlement flow

1.3 – Brokerage fees

1.3.1 – Description

In Mx G2000, up to 4 brokerage fees can be attached to a deal. Brokerage fee details are stored in the table TRN_HDR_DBF, each brokerage fee column being identified by an index (0/1/2/3).

In TRN_HDR_DBF, the source and destination columns M_BRKR_PFL and M_BROKER can contain either a portfolio or a counterparty label.

In Mx.3, there is no limitation on the number of brokerage fees attached to a deal. Brokerage fees are stored in a separate table TRN_BROKER_DBF with n rows per deal number. Therefore, each brokerage fee is identified by the columns M_NB and M_LINE (=0,1,2, ...,n).

In the table TRN_BROKER, the column M_SRC_PFOLIO always contains the ID of the source portfolio. If the source is a counterparty, it is filled with –1. To get the portfolio label, a join with the table TRN_PFLD_DBF is needed. The join should be based on the columns TRN_BROKER_DBF. M_SRC_PFOLIO and TRN_PFLD_DBF.M_LABEL to retrieve the value of the column TRN_PFLD_DBF.M_DSP_LABEL.

The column M_DST_PFOLIO always contains the ID of the destination portfolio. If the destination is a counterparty, it is filled with –1. To get the portfolio label, a join with the table TRN_PFLD_DBF is needed. The join should be based on the columns TRN_BROKER_DBF. M_DST_PFOLIO and TRN_PFLD_DBF.M_LABEL to retrieve the value of the column TRN_PFLD_DBF.M_DSP_LABEL.

The column M_CNTRP contains the ID of the source counterparty if M_SRC_PFOLIO is equal to –1, or the ID of the destination counterparty if the M_DST_PFOLIO is equal to –1. To get the counterparty label, a join with the table TRN_CPDF_DBF is needed. The join should be based on the columns TRN_BROKER_DBF. M_CNTRP and TRN_CPDF_DBF.M_LABEL to retrieve the value of the column.

1.3.2 – Fields Mapping

1.3.2.1 – Data model impact

V2.10 table	V2.10 field	Format in 2.10	V3.1 table	V3.1 table	Format in 3.1
TRN_HDR_DBF	M_BRKR_PFL 0/1/2/3	char(15)	TRN_BROKER_DBF	M_SRC_PFOLIO or M_CNTRP if M_SRC_PFOLIO=–1	numeric(10)
TRN_HDR_DBF	M_BROKER0/1/2/3	char(15)	TRN_BROKER_DBF	M_DST_PFOLIO or M_CNTRP if M_DST_PFOLIO=–1	numeric(10)

TRN_HDR_DBF	M_BRKR_AUTP0/1/2/3	numeric(1)	TRN_BROKER_DBF	M_AUTF	numeric(1)
TRN_HDR_DBF	M_BRKR_AUTS0/1/2/3	numeric(1)	TRN_BROKER_DBF	M_AUTS	numeric(1)
TRN_HDR_DBF	M_BRKR_AVP0/1/2/3	numeric(1)	TRN_BROKER_DBF	M_AVP	numeric(1)
TRN_HDR_DBF	M_BRKR_CODE0/1/2/3	char(3)	TRN_BROKER_DBF	M_CODE	char(10)
TRN_HDR_DBF	M_BRKR_CUR0/1/2/3	char(3)	TRN_BROKER_DBF	M_CUR	char(3)
TRN_HDR_DBF	M_BRKR_FEE0/1/2/3	numeric(16)	TRN_BROKER_DBF	M_FEE	numeric(16)
TRN_HDR_DBF	M_BRKR_TYPE0/1/2/3	numeric(1)	TRN_BROKER_DBF	M_TYPE	numeric(1)
TRN_HDR_DBF	M_BRKR_VTYP0/1/2/3	numeric(1)	TRN_BROKER_DBF	M_VALUE_TYPE	numeric(1)

1.3.2.2 – Dynamic table impact

In TP_ fields, only the first four brokerage fee details of a deal are returned. In dynamic tables of type CS or DT, all brokerage fees are returned in a separate row as usual.

The field TP_BROKER contains an id, to join with the counterpart table.

1.4 – Entities

1.4.1 – Data model schema in Mx.3

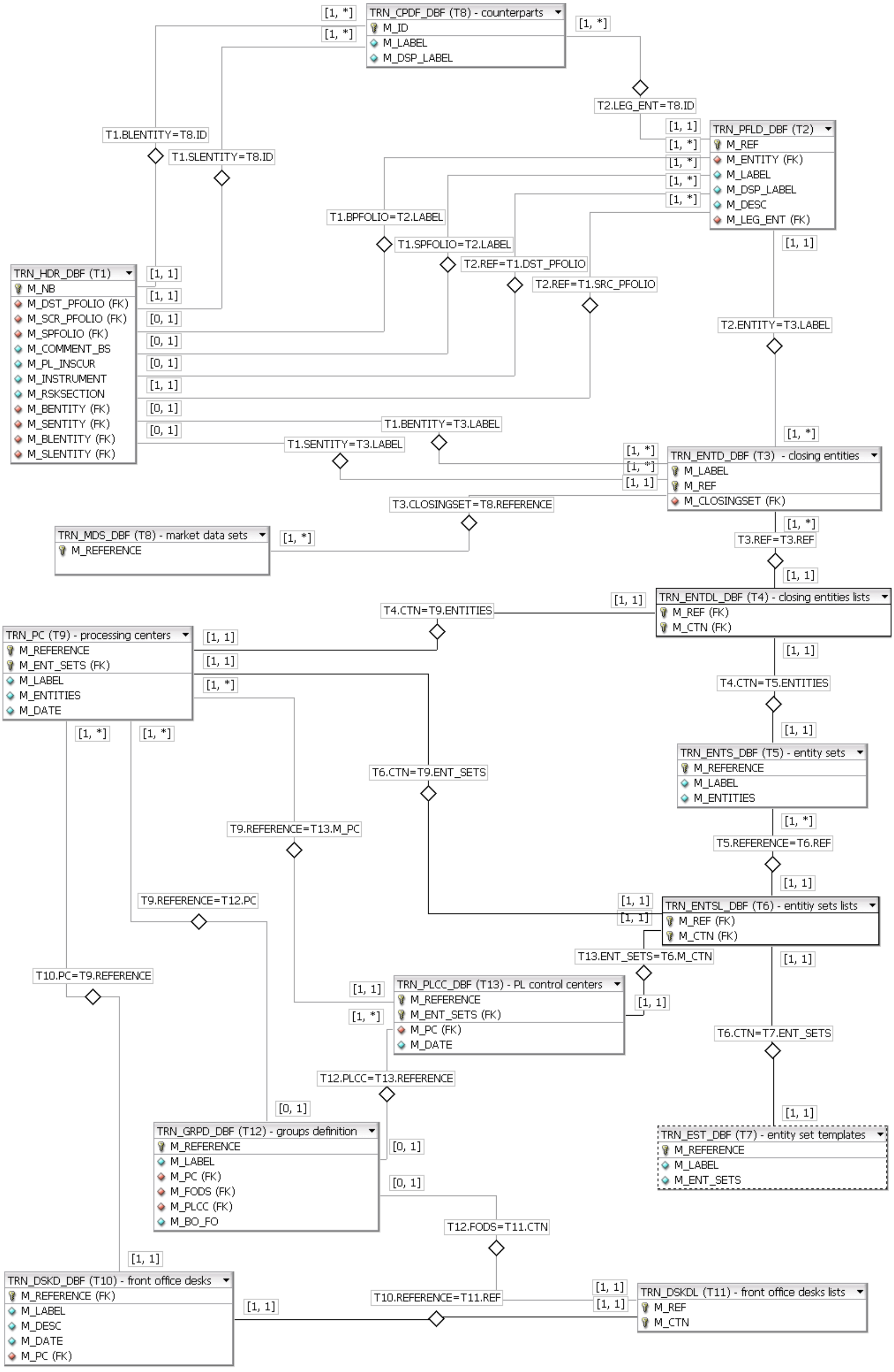

Figure 3: Global Operating model in Mx.3

1.4.2 – Description

In Mx G2000, there is only one type of entity, a portfolio attribute. It is inherited from the portfolio at deal level and can be customised. It is stored in the fields BENTITY and SENTITY in TRN_HDR_DBF (ENTITY for the buy portfolio).

In Mx.3, there are three entity types: the legal entity, the closing entity and the processing entity.

In comparison with 2.10 version, the entity in 2.11 has been enriched with additional features that make it closer to the Mx.3 closing entity.

1.4.2.1 – Closing entity

Definition: Set of portfolios sharing the same end of day close down time and the same market data set. The P&L control group can close down different entity sets at different times and issue an official P&L for each. Closing entities are defined in the portfolio tree used for P&L control (not in 2.11). It is inherited from the portfolio attached to the deal and cannot be customized at deal level. In Mx.3 and recent 2.11 versions, it is not possible to have internal trades between two different closing entities.

Accounting is based on the closing entity. Consequently, information such as the accounting date is stored in the closing entity table TRN_ENTD_DBF.

The fields M_BENTITY and M_SENTITY of TRN_HDR_DBF store in Mx.3 the closing entity of Mx portfolios. Since it is not possible to insert trades between two different closing entities, note that M_BENTITY=M_SENTITY for all trades.

1.4.2.2 – Legal entity

Definition: Reflect the financial institution's effective legal entities or the external legal entities in case of customer servicing. Legal entities are used for settlement instruction assignment and confirmations. Each portfolio can be linked to a legal entity.

Legal entities are stored in Mx.3 in the counterparty table TRN_CPDF_DBF and each portfolio is attached to one legal entity stored in the field M_LEG_ENT of the table TRN_PFLD. Also at trade level, new fields in TRN_HDR_DBF have been created to store the legal entity: M_BLENTITY and M_SLENTITY.

These three fields are filled with the counterparty ID of the entity. The entity label can be looked up by a join with the table TRN_CPDF_DBF, using M_LEG_ENTITY = M_ID to retrieve the label of the legal entity.

Unlike closing entities, it is possible to insert trades between different legal entities.

In pre−filter formulas: To pre−filter on legal entities using labels instead of ids, parser functions can be applied to retrieve the id of the entity from the label. The function CTRP_REF returns the counterparty reference using the display label as parameter.

1.4.3 – Fields mapping

Depending on client implementations, the Mx G2000 entity can be mapped with the closing entity or the legal entity.

V2.10 table	V2.10 field	V3.1 table	V3.1 field
TRN_HDR_DBF	M_BENTITY	TRN_HDR_DBF	M_BENTITY (id of closing entity buy) or M_BLENTITY (id of legal entity buy)
TRN_HDR_DBF	M_SENTITY	TRN_HDR_DBF	M_SENTITY (id of closing entity sell) or M_SLENTITY (id of legal entity sell)

1.5 – Global operating model: dates

1.5.1 – Description

In Mx G2000, global system dates are stored in the same table PROCESS#PS_DATE_DBF and each date can only take one possible value.

In Mx.3, multiple entities can be configured. Consequently, some columns of the table PROCESS#PS_DATE_DBF have been split in different tables and it is now possible to set several dates, for example, different accounting dates per closing entity.

1.5.2 – Fields mapping

V2.10 table	V2.10 field	V2.11 table	V2.11 field	V3.1 table	V3.1 field
PROCESS#PS_DATE_DBF	M_DATE_ACC	TRN_ENTD_DBF	M_ACC_DATE	TRN_ENTD_DBF	M_ACC_DATE
PROCESS#PS_DATE_DBF	M_DATE_PRG	TRN_ENTD_DBF	M_PRG_DATE	TRN_ENTD_DBF	M_PRG_DATE
PROCESS#PS_DATE_DBF	M_DATE_CUT	TRN_ENTD_DBF	M_CUT_DATE	TRN_ENTD_DBF	M_CUT_DATE
PROCESS#PS_DATE_DBF	M_DATE_PL	TRN_PC_DBF	M_DATE	TRN_PC_DBF If one PL control center is configured TRN_PLCC_DBF if more than one PL control center is configured	M_DATE M_DATE
PROCESS#PS_DATE_DBF	M_DATE_FO	TRN_DSKD_DBF	M_DATE	TRN_DSKD_DBF	M_DATE
PROCESS#PS_DATE_DBF	M_DATE_CNS	PROCESS#PS_DATE_DBF	M_DATE_CNS	TRN_ENTD_DBF	M_CNS_DATE

1.5.3 – Migration implementation choice

The view PROCESS#PS_DATE_VW_DBF is provided by Murex to emulate the Mx G2000 data model where dates are unique and stored in one table. It takes arbitrarily the max of the available dates. The corresponding SQL is described in the appendix.

Note that to allow using compatibility views, the default command /DD_COMPATIBILITY in the launcherall.mxres file is mandatory.

The views may faciliatate the Even if u, using views facilitates the migration of the report, it

It is not recommended to use views automatically. Although views do facilitate the migration of the report, they introduce performance problems and complexity to the report.

1.6 – Calendars

1.6.1 – Description

In Mx G2000, calendars are stored in tables CALENDAR_DBF and CLNHDY_DBF. The first table contains calendar definitions with a label, a description, and the weekend days. The secondary table should be used to retrieve holidays attached to each calendar.

Some calendars are defined as unions of calendars whose labels are stored in the columns CALENDAR0, 1... with a maximum of nine calendars.

In Mx G2000, BOCALENDAR is a global variable stored in the table TRN_GCFG_DBF and configured from a Configurator session as general settings in "Calendar back office working days". It returns the back office calendar.

In Mx.3, calendars are stored in the table CAL_DEF_DBF for the calendar definition and the table CAL_HOL_DBF for the holidays.

For unions of calendar, the columns CALENDAR0, 1...9 no longer exist. It is necessary to use the new table CAL_UNI_DBF to retrieve the calendar composition with a 1−n join. This data model enhancement makes it possible to configure unions without any limit on the calendar number. Note that as Mreport does not support 1−n joins, it should be necessary to adapt the report to retrieve all calendars from a union.

In Mx.3, the global variable BOCALENDAR has a different behavior according to the session:

- from a Processing center session, it returns the processing center calendar
- from a P&L control session, it returns the P&L control calendar else the processing center calendar of the P&L control
- from a Front office user session, it returns the FO session calendar else the calendar of the processing center attached to the FO desk.
- from a Configurator session, it returns the back office calendar
- in all sessions, if no calendar is defined, it will always return the back office calendar

The information is stored in the table TRN_DSKD_DBF for the desk calendar and in the table TRN_PC_DBF for the processing center calendar.

In Mx.3, using a missing calendar in a parser function is no longer supported and the system throws an exception with the calendar label.

Note that during the migration phase, a clean up on calendar configuration is performed. If a calendar union references another union, the underlying union is erased. Then if the union is empty, the flag M_ISUNION is forced to zero and the union becomes a regular calendar.

In Mx.3, a new parser function has been developed to retrieve all calendars of a union in one row as in Mx G2000 (note that the order could be different from the Mx G2000). In dynamic tables, the parser function GETUNIONCAL takes as a parameter the calendar name and a separator, and returns the list of underlying calendars. For instance, if the calendar 2EURHKG is formed of the underlying calendars "EUR" and "HKG", the formula GETUNIONCAL(LABEL,';') will return "EUR;HKG;". To retrieve one calendar from the list, the parser function SUBFIELD can then be used (e.g: SUBFIELD(CAL_LIST,1,';')=EUR).

1.6.2 – Data model

1.6.2.1 – In Mx G2000

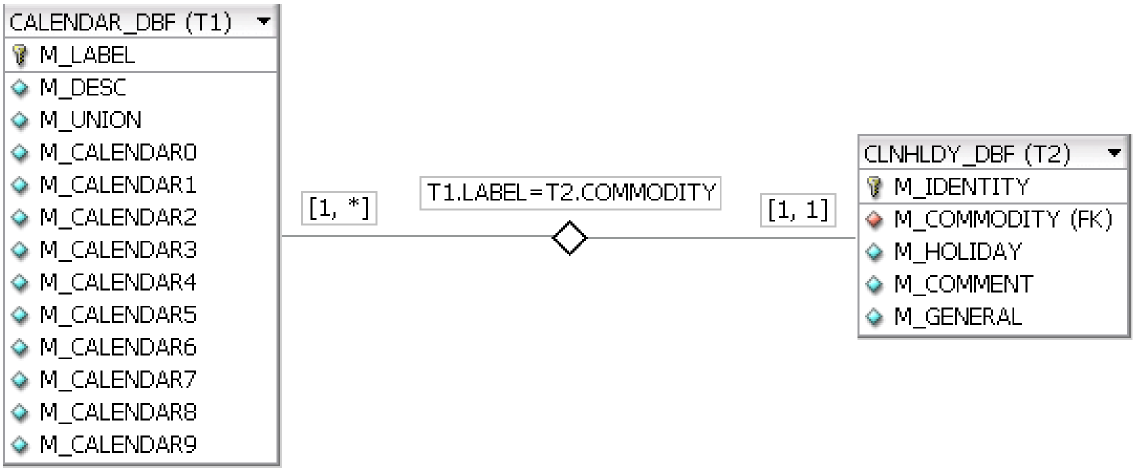

Figure 4: Calendar data model in Mx G2000

1.6.2.2 – In Mx.3

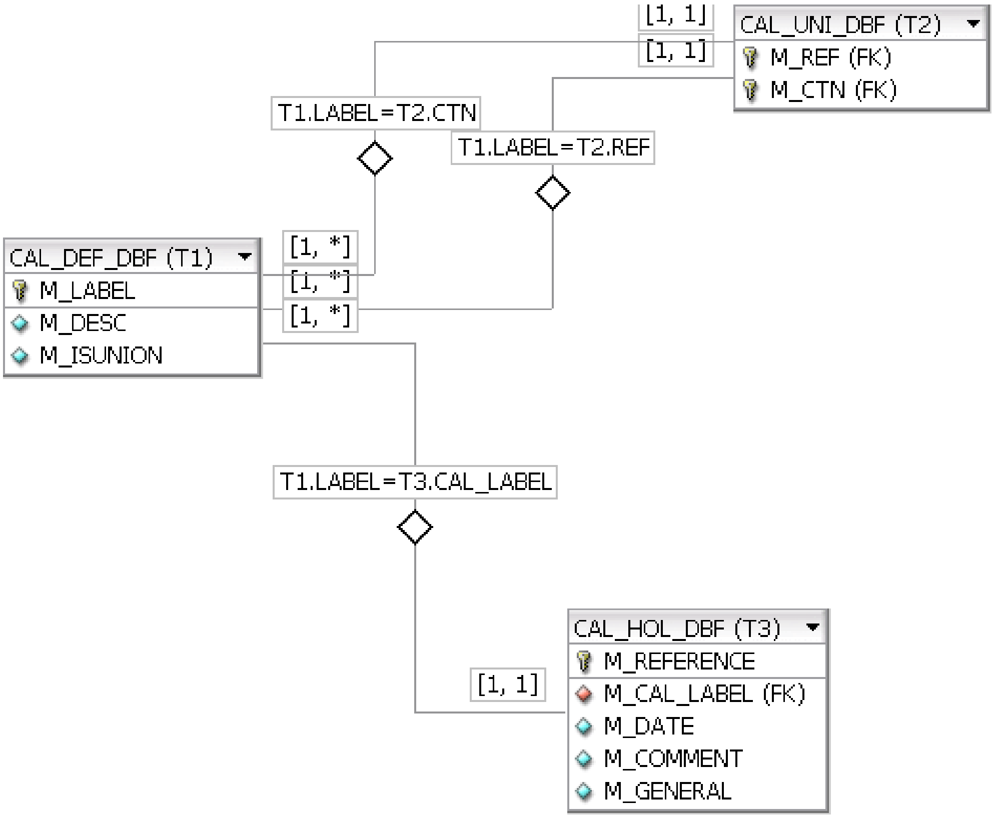

Figure 5: Calendar data model in Mx.3

1.6.3 – Fields mapping

V2.10 table	V2.10 field	Description	V3.1 table	V3.1 field
CALENDAR_DBF	M_LABEL	Calendar label	CAL_DEF_DBF	M_LABEL
CALENDAR_DBF	M_DESC	Calendar description	CAL_DEF_DBF	M_DESC
CALENDAR_DBF	M_SWIFTCODE	Swift code	CAL_DEF_DBF	M_SWIFTCDE
	M_UNION		CAL_DEF_DBF	M_ISUNION

CALENDAR_DBF		union of calendars (Y=yes; N=No)		
CALENDAR_DBF	M_CALENDAR0	Calendar 0 in union	CAL_DEF_DBF	M_REF for CAL_DEF.M_LABEL = CAL_UNI.M_CTN
CALENDAR_DBF	M_CALENDAR1	Calendar 1 in union	CAL_DEF_DBF	M_REF for CAL_DEF.M_LABEL = CAL_UNI.M_CTN
CALENDAR_DBF	M_CALENDAR2	Calendar 2 in union	CAL_DEF_DBF	M_REF for CAL_DEF.M_LABEL = CAL_UNI.M_CTN
CALENDAR_DBF	M_CALENDAR3	Calendar 3 in union	CAL_DEF_DBF	M_REF for CAL_DEF.M_LABEL = CAL_UNI.M_CTN
CALENDAR_DBF	M_CALENDAR4	Calendar 4 in union	CAL_DEF_DBF	M_REF for CAL_DEF.M_LABEL = CAL_UNI.M_CTN
CALENDAR_DBF	M_CALENDAR5	Calendar 5 in union	CAL_DEF_DBF	M_REF for CAL_DEF.M_LABEL = CAL_UNI.M_CTN
CALENDAR_DBF	M_CALENDAR6	Calendar 6 in union	CAL_DEF_DBF	M_REF for CAL_DEF.M_LABEL = CAL_UNI.M_CTN
CALENDAR_DBF	M_CALENDAR7	Calendar 7 in union	CAL_DEF_DBF	M_REF for CAL_DEF.M_LABEL = CAL_UNI.M_CTN
CALENDAR_DBF	M_CALENDAR8	Calendar 8 in union	CAL_DEF_DBF	M_REF for CAL_DEF.M_LABEL = CAL_UNI.M_CTN
CALENDAR_DBF	M_CALENDAR9	Calendar 9 in union	CAL_DEF_DBF	M_REF for CAL_DEF.M_LABEL = CAL_UNI.M_CTN
CALENDAR_DBF	M_WEEKEND0	Sunday week end y/n	CAL_DEF_DBF	M_SUNDAY
CALENDAR_DBF	M_WEEKEND1	Monday week end y/n	CAL_DEF_DBF	M_MONDAY
CALENDAR_DBF	M_WEEKEND2	Thuesday week end y/n	CAL_DEF_DBF	M_TUESDAY
CALENDAR_DBF	M_WEEKEND3	Wednesday – week end y/n	CAL_DEF_DBF	M_WDNESDAY
CALENDAR_DBF	M_WEEKEND4	Thursday – week end y/n	CAL_DEF_DBF	M_THURSDAY
CALENDAR_DBF	M_WEEKEND5	Friday – week end y/n	CAL_DEF_DBF	M_FRIDAY
CALENDAR_DBF	M_WEEKEND6	Saturday – week end y/n	CAL_DEF_DBF	M_SATURDAY
CLNHDY_DBF	M_COMMODITY	Calendar –join CALENDAR_DBF	CAL_HOL	M_CAL_LABEL
CLNHDY_DBF	M_HOLIDAY	Date	CAL_HOL	M_DATE
CLNHDY_DBF	M_COMMENT	Description	CAL_HOL	M_COMMENT

CLNHDY_DBF	M_GENERAL	0 if special holiday, 1 if yearly holiday	CAL_HOL	M_GENERAL

2 – Trades

2.1 – Trade fields

2.1.1 – Description

The trade is described by a set of properties defined in Mx and stored in the table TRN_HDR_DBF. These properties have evolved between Mx G2000 and Mx.3, some are now deprecated whereas others have been added as the trade notion has evolved in Mx.3.

Here is a presentation of Mx.3 enhancements and their impact on the Murex data model.

2.1.1.1 – Impact on market operations of type Replacement (RPL)

In Mx G2000, the fields MRPL_% in TRN_HDR_DBF and in the dynamic tables were used to provide information about the market operations of type RPL such as Restructure, Assignments, Prolongation or Cancel and reissue. When a trade had a MOP of type RPL a new trade number (NB) was generated.

Below is the definition of the different M_MRPL_% fields of TRN_HDR_DBF as they were in Mx G2000:

- M_MRPL_DATE : insertion date of the RPL market operation which is actually the creation date of a trade number (NB), this date is not updated when other market operations (that do not modify the trade number) were applied , and null for the other cases
- M_MRPL_FLAG : equal to 'Y' when the trade was originated by a market operation of type RPL, and 'N' otherwise
- M_MRPL_ONB : populated by :
    - t he initial trade number if the deal is generated by RPL market operation
    - −1 for origin trades (initial M_NB of a trade) which have undergone a mop of type RPL
    - 0 otherwise (no replacement)

In Mx.3, the logic is different, and new fields were introduced to follow the trade's life cycle (last event, origin event, versions etc.). The events of type RPL still generate a new trade number. Most of the MRPL_% fields were kept for backward compatibility:

- M_MRPL_DATE : actual date of the RPL market operation in case of a trade generated by a RPL market operation and actual date of the first version of the trade. Therefore the meaning of M_MRPL_DATE remains the same in Mx.3 which is the actual insertion date of a trade number. The difference in Mx.3 is that this column is never null, it is always populated with the creation date of the version 1 of the trade despite the fact that it was done by a RPL market operation or not.
- M_MRPL_FLAG : removed, deprecated
- M_MRPL_ONB : takes systematically the current trade number and in the case of a trade generated by an RPL market operation it is equal to the trade number of the origin trade on which the market operation was applied. Therefore the logic is still the same as in Mx G2000, the difference being that MRPL_ONB is never null in Mx.3.

In the dynamic tables, the MRPL_% fields follow the same logic as their equivalent in TRN_HDR_DBF in both Mx G2000 and Mx.3.

Since MRPL_FLAG is no longer used in Mx.3, the condition MRPL_ONB <> NB can be used to identify the trade that has been subject to a market operation RPL.

When migrating the database from Mx G2000 to Mx.3, M_MRPL_ONB is populated by M_NB in TRN_HDR_DBF for the trades where M_MRPL_ONB was <= 0.

Impact on migration: The field M_MRPL_FLAG should not be used anymore. Only the MRPL_DATE field is used as it is filled for all trades.

*Applications, f*rom Mx G2000 to Mx.3:

- Sample 1:
  - In Mx G2000:

    ```
    select count(*) from TRN_HDR_DBF where M_MRPL_ONB=−1
    ```

  - In Mx.3:

    ```
    select count(*) from TRN_HDR_DBF A where A.M_NB in (select B.M_ORIG_TRADE from TRN_EXT_DBF B
    where B.M_ORIG_TRADE=A.M_NB and (B.M_ORIG_TRADE<>B.M_TRADE_REF or B.M_EVT_INTID='1.220')
    and M_ACTION<>8)
    ```

- Sample 2: to reproduce in Mx.3 MRPL_ONB as in Mx G2000

```
select

case when A.M_MRPL_ONB <> M_NB then A.M_MRPL_ONB

else

(case when ((select count(*) from TRN_HDR_DBF T2 where T2.M_MRPL_ONB = A.M_MRPL_ONB)=1 and (select
count(*) from TRN_EXT_DBF EXT where M_ORIG_TRADE = A.M_MRPL_ONB and M_EVT_INTID = '1.220') = 0)

then 0 else −1 end) end as M_MRPL_ONB

from TRN_HDR_DBF A
```

2.1.1.2 – Impact on trade acceptance

With the new concept of trade acceptance in Mx.3, it is possible to retrieve only the last accepted version of a contract in the report/extraction:

- Using SQL:
  - In manual mode: By default, for a new version the M_LAT field is set to 1. When the version is accepted, the flag is set to 0 and the actual date is updated (only for the first version of the trade in TRN_HDR_DBF and for all versions in TRN_EXT_DBF) . The query should retrieve from TRN_EXT_DBF the max (M_VERSION) where M_LAT = 0.
  - In implicit mode: The flag M_LAT is not used (by default, M_LAT = 0) whereas the acceptance is based on the actual date. The query should retrieve from TRN_EXT_DBF the max (M_VERSION) where M_ACT_DATE <= PC Date (Processing center date).
- With dynamic tables: a PL Control group should be used.

2.1.1.3 – Contract workflow impact on Validation status

In Mx G2000, the trade validation status M_VAL_STATUS was stored in the table TRN_HDR_DBF.

In Mx.3, the trade validation status is set by the contract work flow and stored at the level of the table CONTRACT_DBF in the field M_STP_STATUS. In dynamic table pre−filters, the filter formula should be updated accessing the table CONTRACT_DBF through the union to retrieve the validation status.

Migration impact: The validation status should be retrieved from the CONTRACT_DBF table. In dynamic table pre−filter, the filter formula M_VAL_STATUS = 'XXXXX' should be replaced by an access through the contract table with the following formula CONTRACT−>STP_STATUS= 'XXXXX'.

2.1.1.4 – Trade repository impact on linked trades structure

The linked trade notion was implemented in Mx G2000 to link trades together. The table LTI#LTI_HDR_DBF stores the linked trade characteristics and the table LTI#PAC_TMPL stores the typology of these linked trades.

In Mx.3, this logic has been re−engineered. Contracts are used to group contract components sharing the same counterparty, and packages regroup contracts having different counterparties. The migration procedure transforms existing Mx G2000 linked trades into contracts or packages (linked trades with the same counterparty across legs and issued from STB public strategies are migrated into contracts). The tables CONTRACT_DBF and PACKAGE_DBF are updated accordingly.

Migration impact: The fields located in LTI#LTI_HDR_DBF table can be retrieved using the tables PACKAGE_DBF and TYPOLOGY_DBF.

When defining a linked trade in Mx G2000, it was possible to configure a main trade. The main trade (M_MAIN_NB) could be used to distinguish on which trade the accounting notion of "off balance sheet " was based on.

In Mx.3, it is no longer possible to define a component as main trade when defining a contract or a package. The main trade concept may be reintroduced in future releases.

Migration impact: The main trade is deprecated.

2.1.1.5 – Trade repository impact on trade typology

In Mx.3, the typology can be defined at each level: trades, contracts or packages. These typologies are stored in the same table TYPOLOGY_DBF.

The parser function FLOW_TYPO returns (given the flow typology ID and the column index) the flow sub−typology contained in the table TYPOLOGY_DBF. In the example above, FLOW_TYPO(12,0) returns CAP

Migration impact: Depending on client implementation choices, the typology should be retrieved at the level of the trade, the contract or the package.

2.1.1.6 – Trade versioning impact on trade status

In Mx G2000, the field TRN_STATUS used to return:

- LIVE: deal inserted or created by a Market operation (MOP)
- MKT_OP: deal partially closed
- DEAD: dead (expired or canceled)

In Mx.3, the field TRN_STATUS behaves differently (see mapping table below) and MKT_OP means deal modified by an event.

In dynamic tables, TP_STATUS follows the same logic as the field TRN_STATUS in TRN_HDR_DBF. It is no longer recommended to use it.

The field AMD_STS is an enrichment of the field TP_STATUS to cover the new Mx.3 concepts such as the trade acceptance, the versioning. When possible, use this field from the dynamic table instead of accessing directly the data model because TRN_STATUS is not contextual.

AMD_STS returns :

- O if the deal is live (or open)
- OA (or open amended) if the deal has been modified by an event or a fixing, or deal created by an event (like allocation, step in for example)
- CL (closed) if the deal is expired or has been subject to a full unwind
- CA (canceled) if the deal has been subjected to a Cancel event
- NA (Not Available) if the trade is not available for the computing parameters (date, group for instance).

The corresponding mapping is described in the table below:

TP_STATUS in Mx G2000	Description	TP_STATUS in Mx.3	AMD_STS in Mx.3	Description
TP_STATUS = LIVE	Inserted trades	TP_STATUS = LIVE	AMD_ STS = O	Inserted trades
TP_STATUS = LIVE	New deal created by a MOP of type RPL like Restructure deal/assignment/etc	TP_STATUS=LIVE	AMD_STS=OA	Deal modified by a market op if version > 1 Or New deal created by a mop if version=1
TP_STATUS = DEAD	Trade on which a MOP of type RPL (C&R or restructure) has been applied	TP_STATUS =DEAD	AMD_STS=NA	If trade has been replaced (C&R or restructure)
TP_STATUS = DEAD	Deal dead (full early termination, exercise)	TP_STATUS=DEAD	AMD_STS=CL	Closed deal (full unwind, full step out, exercise)
TP_STATUS = DEAD	Deal dead (cancel)	TP_STATUS=DEAD	AMD_STS=CA	Deal cancelled (by contract event Cancel)
TP_STATUS = MKT_OP	Deal partially dead, impacted (mop that does not impact the trade number and does not close the deal (partial early termination etc))	TP_STATUS=MKT_OP	AMD_STS=OA	Deals modified by an event that does not modify the trade number and does not close the deal like partial unwind
TP_STATUS=''	If deal is not Available for the computing date (if MRPL date > computing date)	TP_STATUS=''	AMD_STS=NA	If deal is not Available for the computing date or for the user/group running the computation (for e.g. if deal actual date > computing date or extraction run by a pl control group on a deal not accepted in the PL). It's also the status for all previous versions of the deal

Migration impact: The field TP_STATUS from the dynamic tables should be replaced by the field AMD_STS from the dynamic tables.

Consequence on filters:

Pre−filters: The pre−filter TRN_STATUS='LIVE' in Mx G2000 should be migrated as TRN_STATUS<>'DEAD' as trades impacted by restructures in Mx.3 have their status set to 'LIVE' instead of 'DEAD'.

Post−filters: To reproduce the Mx G2000 behavior of the dynamic table field TP_STATUS2 in Mx.3, it is possible to build a horizontal field OLD_STATUS with the following formula:

```
IIF(AMD_STS2='NA'.AND.MRPL_DATE>DENV('CRT_BND12'),'',IIF(AMD_STS2='NA','DEAD',IIF(AMD_STS2='OA'.AND.(CNTLEV
```

The post–filter TP_STATUS2='LIVE' becomes OLD_STATUS='LIVE'

2.1.1.7 – Impact on other trade fields

Draft deals and field M_REAL

In Mx G2000, the field M_REAL of TRN_HDR_DBF used to return 1 if the deal was draft, 2 if it was real, and 3 if it was committed.

In Mx.3, the distinction between real and committed doesn't exist since the deals are consolidated in real–time at deal insertion. Therefore, the column M_REAL is deprecated, all "real" deals are automatically "committed", and drafts deals are identified by the column M_PURPOSE. More specifically, the column M_PURPOSE can be filled at the trade insertion with Draft, MMDA or Cash Transfer.

In dynamic tables, the same logic applies. The column TP_REAL is deprecated, and the purpose of the trade is returned by the column PURPOSE.

For a customer with no draft deals, the information is deprecated.

Front office validation and field AGREED_TRN

In Mx G2000, the field AGREED_TRN returned 'Y' or 'N' whether the deal has been validated or not by the Front–Office.

In Mx.3, this field is no longer used and the deal validation is handled by the STP workflows which are client specific. Therefore the logic of the field AGREED_TRN can be replicated in Mx.3 via the field STP_STATUS in the dynamic tables or the field STATUS in the CONTRACT_DBF table that holds the contract current status in the workflow.

2.1.2 – Data model schema

2.1.2.1 – In Mx G2000

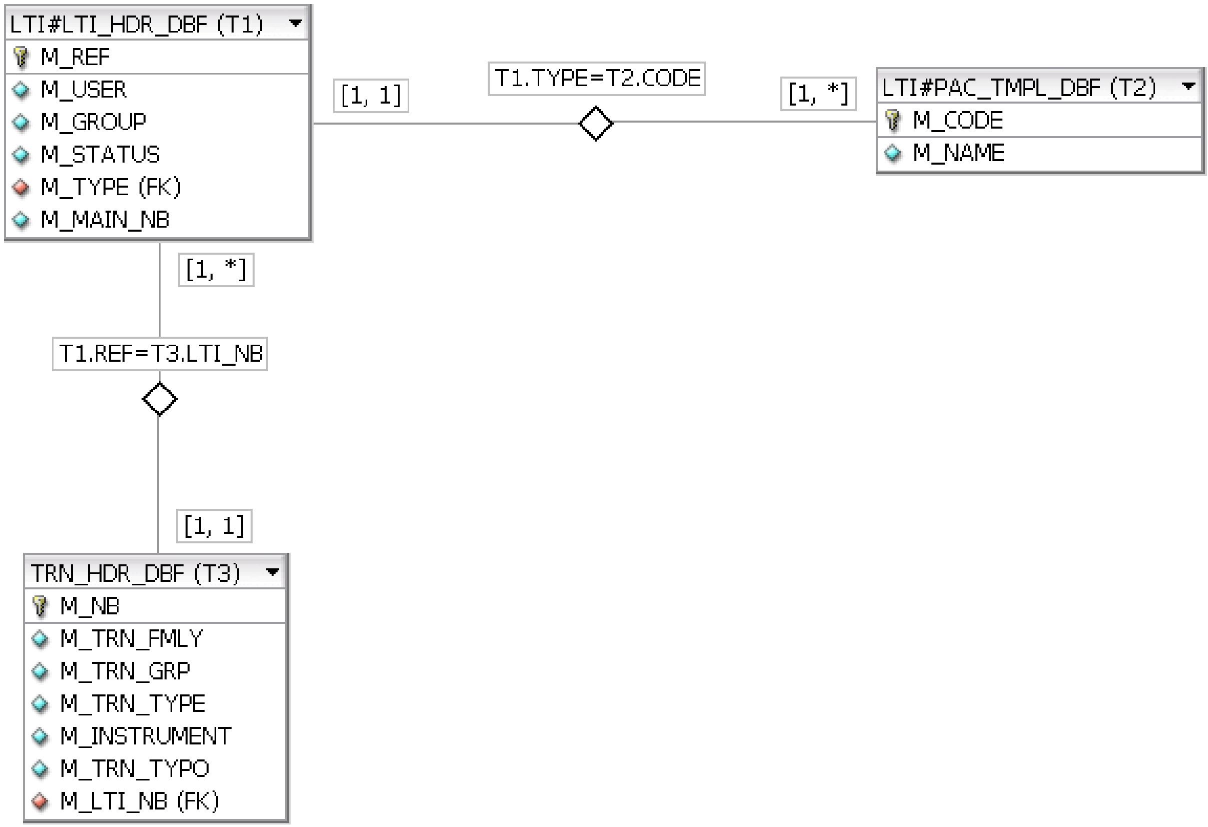

Figure 6: Link between TRN_HDR_DBF and linked trades table (v2.10)

2.1.2.2 – In Mx.3

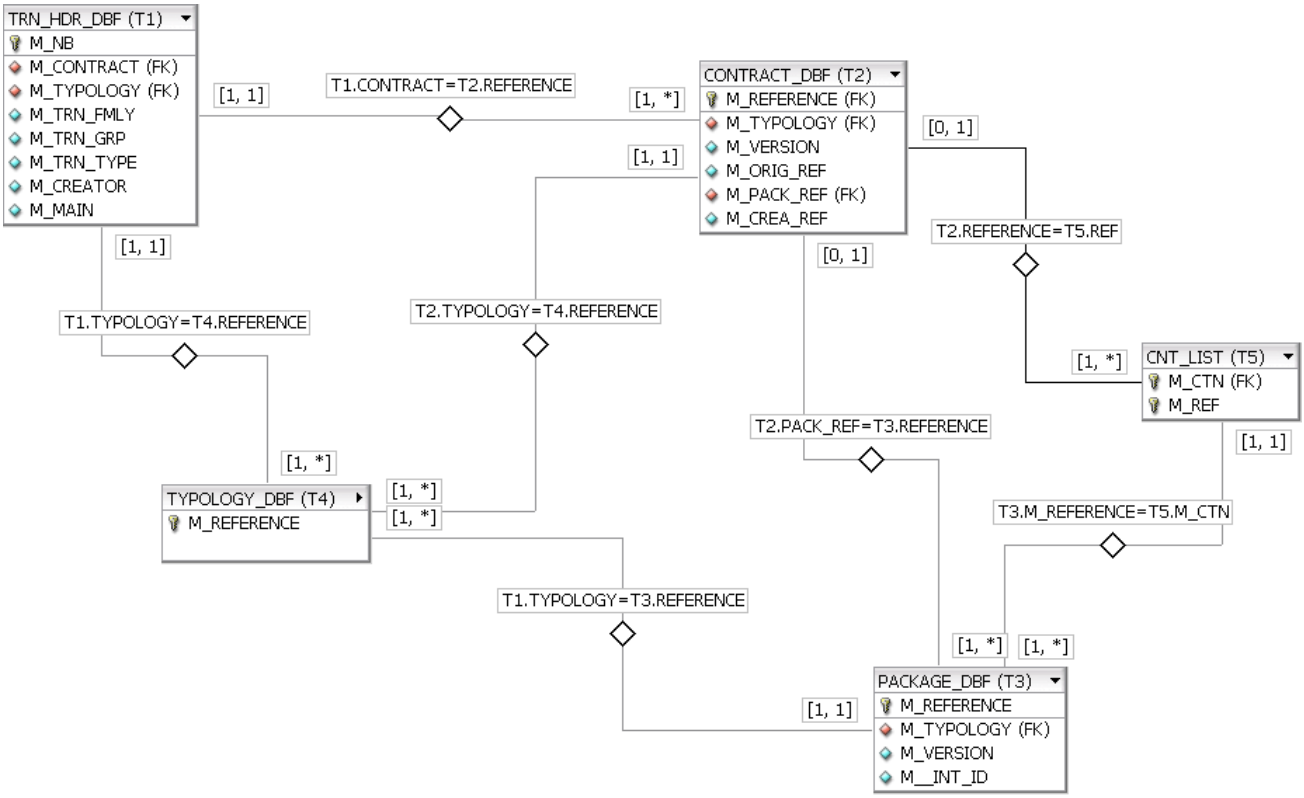

Figure 7: Link between TRN_HDR_DBF, contracts and packages (v3.1)

2.1.3 – Fields Mapping

2.1.3.1 – Data model impact

V2.10 table	V2.10 field	V3.1 table	V3.1 field	Description
TRN_HDR_DBF	M_MRPL_DATE	TRN_HDR_DBF	M_MRPL_DATE	Actual date of the insertion of the trade number
TRN_HDR_DBF	M_MRPL_FLAG	Not maintained		
TRN_HDR_DBF	M_MRPL_ONB	TRN_HDR_DBF	M_MRPL_ONB	Origin trade number
TRN_HDR_DBF	M_BPFOLIO	TRN_HDR_DBF	M_BPFOLIO	Portfolio / counterparty sell
TRN_HDR_DBF	M_SPFOLIO	TRN_HDR_DBF	M_SPFOLIO	Portfolio / counterparty buy
TRN_HDR_DBF	M_COMMENT_BS	TRN_HDR_DBF	M_COMMENT_BS	Buy or Sell
None	None	TRN_HDR_DBF	M_DST_PFOLIO	Destination portfolio id
None	None	TRN_HDR_DBF	M_SRC_PFOLIO	Source portfolio id
None	None	TRN_HDR_DBF	M_COUNTRPART	Destination counterparty id
TRN_HDR_DBF	M_VAL_STATUS	CONTRACT_DBF	M_STP_STATUS	Validation status
TRN_HDR_DBF	M_LTI_NB	PACKAGE_DBF	M_REFERENCE	Contract number
TRN_HDR_DBF	M_MAIN	Not maintained	deprecated	Waiting for new developments, NB instead
LTI#LTI_HDR_DBF	M_REF	PACKAGE_DBF	M_REFERENCE	Package reference
LTI#LTI_HDR_DBF	M_MAIN_NB	Not maintained		
LTI#LTI_HDR_DBF	M_TYPE	PACKAGE_DBF	M_TYPOLOGY	
LTI#PAC_TMPL_DBF	M_CODE	TYPOLOGY_DBF	M_LABEL	join to retrieve the label of the

				package typology: TYPOLOGY.M_REFERENCE = PACKAGE.TYPOLOGY
LTI#PAC_TMPL_DBF	M_NAME	TYPOLOGY_DBF	M_DESC	join to retrieve the description of the package typology: TYPOLOGY.M_REFERENCE = PACKAGE.TYPOLOGY
TRN_HDR_DBF	M_TRN_STATUS	TRN_HDR_DBF	M_TRN_STATUS	Trade status
TRN_HDR_DBF	M_REAL	Deprecated, use TRN_HDR_DBF	M_PURPOSE	
TRN_HDR_DBF	M_AGREED_TRN	CONTRACT_DBF	M_STATUS	Status from the workflow

2.1.3.2 – Dynamic table impact

V2.10 fields	V2.10 description	V3.1 fields	V3.1 description
MRPL_DATE	Replacement date (only filled for RPL MOP)	MRPL_DATE	Replacement date = Actual date (always filled) of the insertion of the trade number
MRPL_FLAG	Replacement flag (equal to Y for RPL MOP)	Not maintained	Deprecated
MRPL_ONB	Replacement number (only filled for RPL MOP)	MRPL_ONB	Origin trade number (always filled)
NB_INIT	Replacement number for RPL MOP else trade number (always filled)	NB_INIT	Same behavior NB_INIT = MRPL_ONB
TP_VALSTAT	Validation status	STP_STATUS	Last validation status
TP_NBLTI	LTI number	PACKAGE	Package reference
TP_TYPO	Trade typology	Depending on client implementation: • CMP_TYPO and/or • CNT_TYPO and/or • PKG_TYPO	Depending on client implementation: • Trade typology • Contract typology • Package typology
TP_MAIN	Main trade	Not maintained	
TP_STATUS	Trade status	AMD _STATUS	Trade status
TP_REAL	If drattf deal TP_REAL = 1	PURPOSE	If draft deal PURPOSE = "Draft'

2.2 – Trade User definable fields

2.2.1 – Description

The trade is described in MX by a set of properties and stored in TRN_HDR_DBF. These predefined properties can be extended and customized by user definable fields (UDF) stored in TABLE#DATA#DEAL%_DBF tables. There are several UDF tables depending on the family of the trade.

In Mx G2000 version, the link between TRN_HDR_DBF and user definable fields is a 1–1 join as they were not versioned.

In Mx.3, trades are versioned and consequently user definable fields also. Every modification to the UDF fields generates a new row with a new reference in the UDF tables. A new version of a deal generates a new row in the UDF table only if UDF have been modified or if the event is a Cancel and reissue.

2.2.1.1 – Data model

All trade versions are stored in the table TRN_EXT_DBF. To access expected versions of user definable fields, a link between the field M_UDF_REF in the table TRN_EXT_DBF and the field M_NB of UDF tables, identifier of the UDF record, can be used. For a given version of the deal in TRN_EXT_DBF, it is possible to retrieve the corresponding user definable fields in TABLE#DATA#DEAL%_DBF and the corresponding trade characteristics in TRN_HDR_DBF.

For a given record in TRN_HDR_DBF, it is possible to have access to all trade versions in TRN_EXT_DBF. To retrieve information for an expected version, this join is not sufficient as the trade version needs to be specified. The most recent version of user definable fields can be retrieved with direct access to MX data model, using either the version of the contract in the table CONTRACT_DBF, which corresponds to the last version, or considering only the last record in TRN_EXT_DBF for each trade (group by clause based on M_TRADE_REF). Both solutions give the same result as in v2.10, i.e. only the last version of user definable fields for a given record in TRN_HDR_DBF is extracted.

2.2.1.2 – Dynamic tables

In dynamic tables, new fields have been added to facilitate the join between the dynamic tables and the UDF tables. The reference of the UDF available for a trade version is given by the following fields:

UDF_REF in TRNRP_PL

V_UDF_REF in TRNRP_MK

2.2.2 – Datamodel schema

2.2.2.1 – In Mx G2000

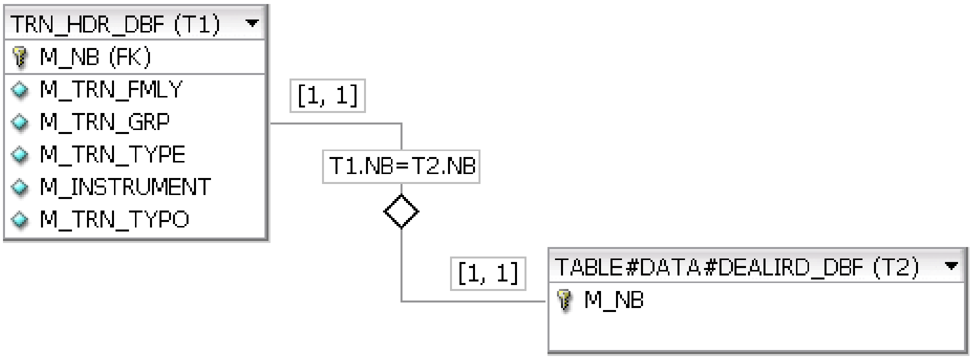

Figure 8: Link between TRN_HDR_DBF and user definable fields (2.10)

2.2.2.2 – In Mx.3

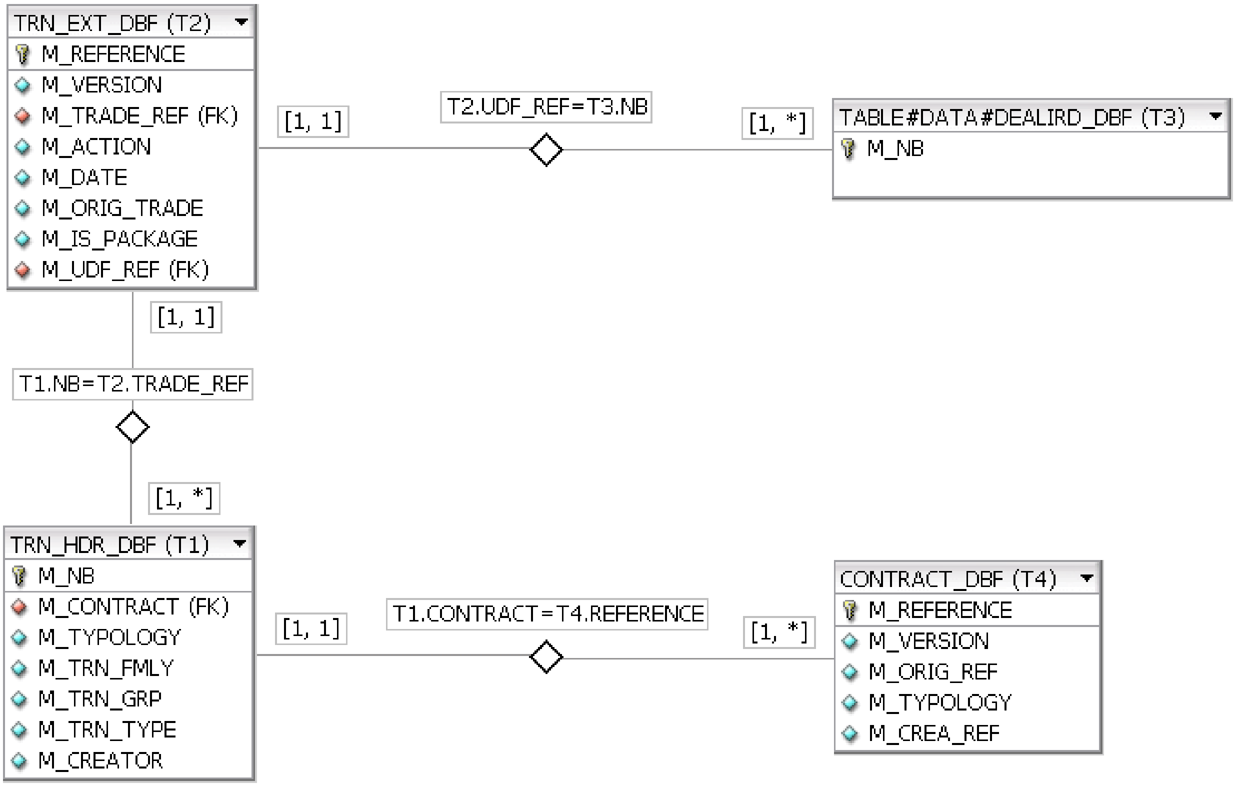

Figure 9: Link between TRN_HDR_DBF and user definable fields (3.1)

2.2.3 – Fields Mapping

2.2.3.1 – Migration implementation choice

Views are provided by Murex to emulate a simple link between TRN_HDR_DBF and the UDF tables TABLE#DATA#DEALxxx_DBF. There is one view per UDF table using the following naming convention: TABLE#DATA#DEALxxx_VW_DBF. They are created and maintained automatically by MX.

Note that to allow using compatibility views, the default command /DD_COMPATIBILITY in the launcherall.mxres file is mandatory.

It is not recommended to use views automatically. Although views do facilitate the migration of the report, they introduce performance problems and complexity to the report.

The corresponding SQL is described in the Appendix UDF views.

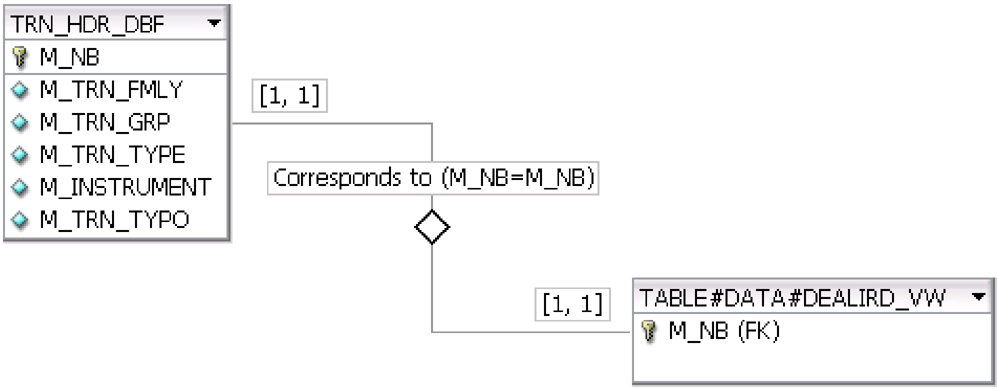

Figure 10: Link between TRN_HDR_DBF and view on user definable fields (3.1)

2.2.3.2 – Data model impact

V2.10 table	V2.10 field	V3.1 table	V3.1 field
TABLE#DATA#DEALIRD_DBF	Fields	TABLE#DATA#DEALIRD_VW_DBF	Fields
TABLE#DATA#DEALCURR_DBF	Fields	TABLE#DATA#DEALCURR_VW_DBF	Fields
TABLE#DATA#DEALEQD_DBF	Fields	TABLE#DATA#DEALEQD_VW_DBF	Fields
TABLE#DATA#DEALCRD_DBF	Fields	TABLE#DATA#DEALCRD_VW_DBF	Fields
etc...		etc...	

2.3 – Cash Flows

In Mx G2000 and in Mx.3, the table TRN_HDRF_DBF contains the transaction flows.

2.3.1 – Flow typologies

2.3.1.1 – Description

Flow typologies can be defined in Mx through five categories.

In Mx G2000, the flow typology is stored in the table TRN_HDRF_DBF thanks to two different fields: M_FLOW_TPT(0/1/2/3/4) and M_FLOW_TPL(0/1/2/3/4). The first set of fields indicates the category and the second set of fields gives the label of the category.

In Mx.3, typologies still have five categories but they are stored in a different way. In the table TRN_HDRF_DBF, each flow has a typology identifier with the field M_FLOW_TYPEID. This field can be used to join with the new table FLOW_TYPO_DBF to retrieve the label of the five typologies in the fields M_TYPE0/1/2/3/4. Note that in this case, one only field allow to understand the five flow typologies.

Here is an example:

A flow (reference = 20) has the following typology:

Category 0: CAP Category 2:

Category 1: STLCategory 3: BEGCategory 4:

In Mx G2000, in TRN_HDRF_DBF:

NB	FLOW_ TPT0	FLOW_ TPT1	FLOW_ TPT2	FLOW_ TPT3	FLOW_ TPT4	FLOW_ TPL0	FLOW_ TPL1	FLOW_ TPL2	FLOW_ TPL3	FLOW_ TPL4
20	Categ0	Categ1	Categ3			CAP	STL	BEG		

In Mx.3, in TRN_HDRF_DBF :

NB	FLOW_TYPEID
20	12

and in FLOW_TYPO_DBF

FLOW_TYPEID	TYPE0	TYPE1	TYPE2	TYPE3	TYPE4
12	CAP	STL		BEG	

2.3.1.2 – Fields mapping

2.3.1.2.1 – Data model impact

V2.10 Table	V.2.10 field	V3.1 Table	V3.1 Field	Comments
TRN_HDRF_DBF TRN_HDRF_DBF	M_FLOW_TPT0 M_FLOW_TPL0	FLOW_TYPO_DBF	M_TYPE0	Use the field M_FLOW_TYPID from TRN_HDRF_DBF to retrieve the typologies
TRN_HDRF_DBF TRN_HDRF_DBF	M_FLOW_TPT1 M_FLOW_TPL1	FLOW_TYPO_DBF	M_TYPE1	Use the field M_FLOW_TYPID from TRN_HDRF_DBF to retrieve the typologies
TRN_HDRF_DBF TRN_HDRF_DBF	M_FLOW_TPT2 M_FLOW_TPL2	FLOW_TYPO_DBF	M_TYPE2	Use the field M_FLOW_TYPID from TRN_HDRF_DBF to retrieve the typologies
TRN_HDRF_DBF TRN_HDRF_DBF	M_FLOW_TPT3 M_FLOW_TPL3	FLOW_TYPO_DBF	M_TYPE3	Use the field M_FLOW_TYPID from TRN_HDRF_DBF to retrieve the typologies
TRN_HDRF_DBF TRN_HDRF_DBF	M_FLOW_TPT4 M_FLOW_TPL4	FLOW_TYPO_DBF	M_TYPE4	Use the field M_FLOW_TYPID from TRN_HDRF_DBF to retrieve the typologies

2.3.1.2.2 – Dynamic table impact

In the dynamic table TRNRP_CS, the set of fields F_TYPELAB(0/1/2/3/4) return the different typologies. These fields have the same behavior in Mx G2000 as in Mx.3 but do not cover the user defined typologies that only exist in Mx.3.

To manage all typologies (including the user defined ones), a new set of dynamic table fields are available in Mx.3: F_TYPO (0/1/2/3/4).

3 – Events

3.1 – Data model schema

3.1.1 – In Mx G2000

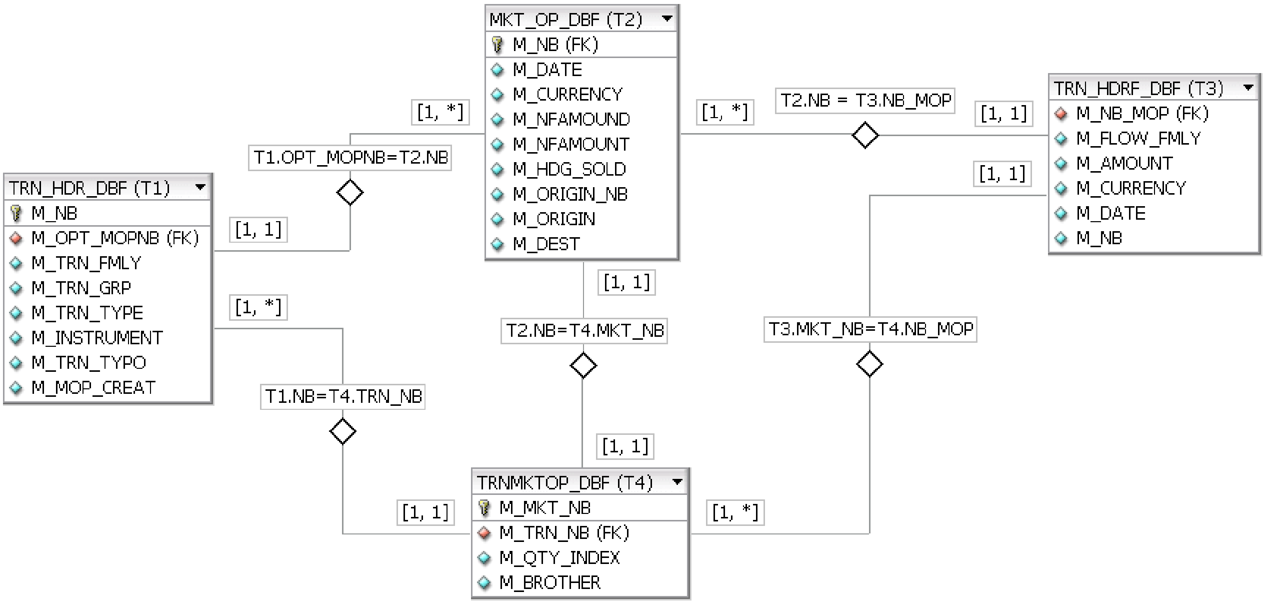

Figure 11: Link between TRN_HDR_DBF and Market operations (v2.10)

3.1.2 – In Mx.3

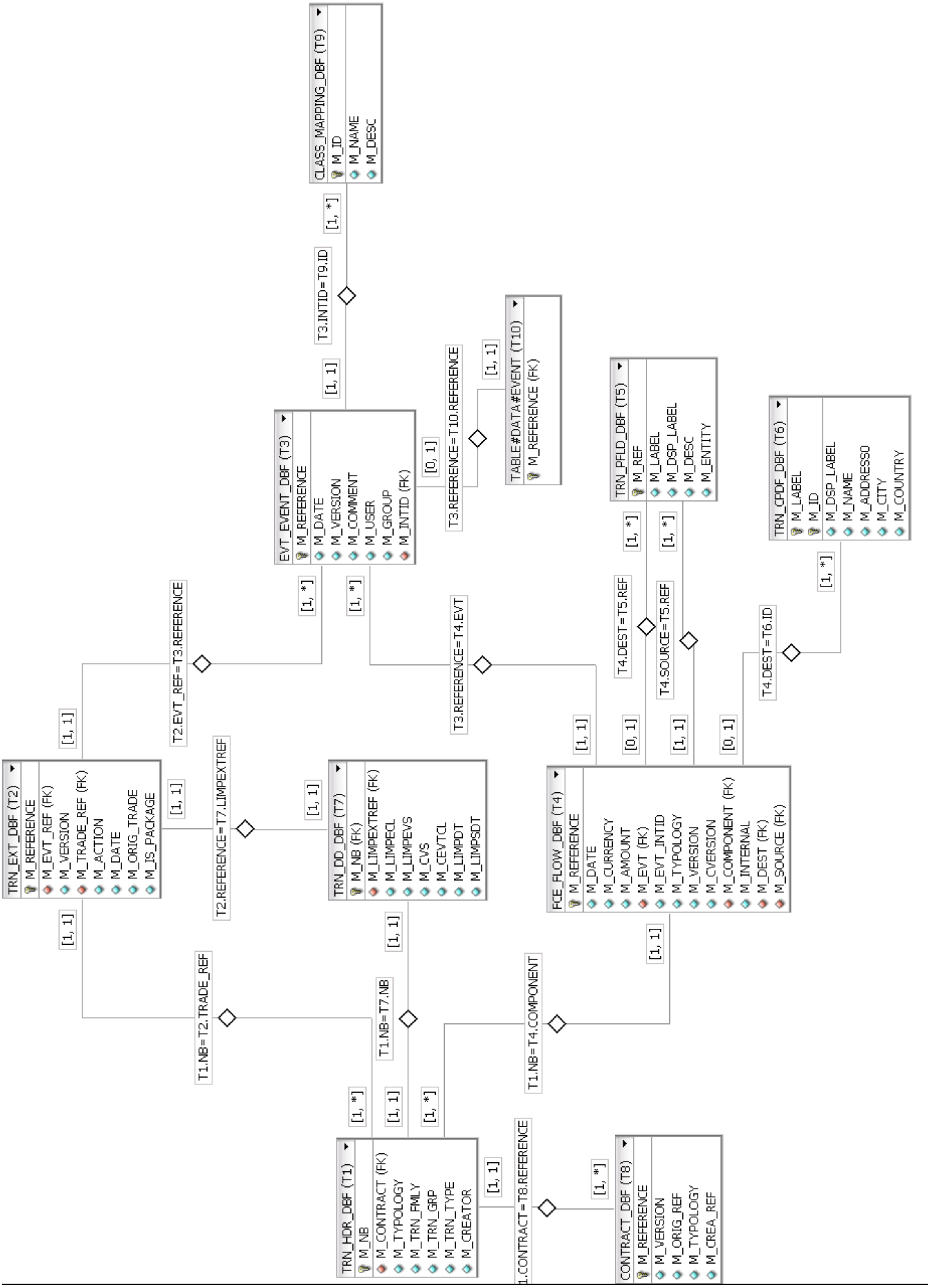

Figure 12: Events data model in MX III

3.2 – Market operations

3.2.1 – Description

In Mx G2000, trades can be modified by market operations. For each trade in TRN_HDR_DBF, it was possible to retrieve information about the associated market operation in the tables TRMKTOP_DBF (header) and MKT_OP_DBF (body). Note that in TRN_HDR_DBF, the field M_OPT_MOPNB was a shortcut that allowed a 1–1 join with the table MKT_OP_DBF to retrieve the last market operation.

In Mx.3, market operations are called *events*. Each time an event is performed, trades are impacted and versioned in the table TRN_EXT_DBF. For each version of a trade, event information can be retrieved in the table EVT_EVENT_DBF (as well as events performed on payments – deliverable cash). In the table TRN_EXT_DBF, there is one record per trade per version whereas in the table EVT_EVENT_DBF, there is one record per event. In this data model, there is a 1–N join between TRN_HDR_DBF and TRN_EXT_DBF. To retrieve the information on the last event, it is necessary to have the last trade version that can be retrieved from the table CONTRACT_DBF or considering only the last record of each trade in TRN_EXT_DBF.

3.2.1.1 – Market operation type

The naming convention has changed between Mx G2000 and Mx.3, and also new events are available in Mx.3. Note that in Mx.3, a new table CLASS_MAPPING_DBF has been created and contains the mapping between the event reference and the event label.

Below, the list of market operation types, their labels in Mx G2000 and the correspondence in Mx.3:

Typology label in 2.10	Sub type in 2.10	Typology reference in 2.10	Description in 2.10	Typology label in 3.1	Typology reference in 3.1	Description in 3.1
RPL	ASSIG	5	Assignment	mxContractEventIRESTRUCTURE	1.373	Restructure
RPL_D		7	Cancel	mxContractEventICANCEL	1.220	Cancel
RPL_M		6	Cancel and Reissue	mxContractEventICANCEL_REISSUE	1.372	Cancel and Reissue
XIT		3	Early Termination	mxContractEventIUNWIND	1.371	Unwind (Termination)
EXR		1	Exercise	mxContractEventIEXERCISE	MwDcj67841	Exercise
EXP		2	Expiry	mxContractEventIEXPIRY	1.589	Expiry
NET		4	Netting	mxContractEventINETTING	MJgYK37904	Netting
RPL	PROL	5	Prolongation	mxContractEventIRESTRUCTURE	1.373	Restructure
RPL		5	Restructure	mxContractEventIRESTRUCTURE	1.373	Restructure

Note that the migration script does not take into account the sub typology. Consequently, the market operation RPL–ASSIG and RPL–PROL will be migrated as a restructure.

3.2.1.2 – Fields in TRN_HDR_DBF

Some fields were available in TRN_HDR_DBF like M_OPT_MOPNB, M_OPT_MOPLST to facilitate joins between tables or to allow their use in pre–filter formulas. All these fields are not maintained in Mx.3 and they have been shifted into another table.

In Mx.3, a new table TRN_DD_DBF can be used as an extension of the table TRN_HDR_DBF. The join between this table and TRN_HDR is a 1−1 join on the trade number. The TRN_DD table facilitates the join with the table TRN_EXT_DBF as it allow a 1−1 join with TRN_EXT_DBF without using the CONTRACT_DBF table.

Note that to allow the use of this compatible table TRN_DD_DBF by MX, it is necessary to add the default command /DD_COMPATIBILITY in the launcherall.mxres file.

An other important goal of this table is to be accessed by pre−filter formulas and to allow migration of pre−filters.

The mapping between old and new fields is described in the "Fields Mapping " sub−section.

3.2.1.3 − Bundles migration

In MxG2000, several events (like extension, early take up, MM roll over etc...) were not considered as MOPs. These events are related to the bundles concept in MxG2000.

The following tables are no longer used in Mx.3

BUNDLE#BODY_DBF

BUNDLE#AUDIT_B_DBF

BUNDLE#HEAD_DBF

BUNDLE#AUDIT_H_DBF

Bundle components in Mx G2000 (MOP and trades) will be migrated as follow:

- MOP like extension, expiry will be migrated as event
- Trades will be migrated as contracts with one expiry for example.
- An event "hold" has been created to emulate the lock produced by the bundle in Mx G2000.
- The link between bundle components will be lost.
- Trade that has been locked (when the field M_STS_FLAG in TRN_HDR_DBF is different than 0, the deal is locked) will be impacted by a specific event. The flag in TRN_HDR is deprecated as it has been replaced by an event.

3.2.1.4 − Removed market operation

In Mx G2000, market operations were not versioned either. It was only possible to insert a mop or to delete it (removed mop). When canceling a market operation in Mx G2000, the market operation (and resulted trade for restructure and cancel and reissue) was physically deleted from the market operation tables.

In Mx.3, deleting a mop is no more available as it is possible to use the versioning. Indeed in Mx.3, actions on events can be performed, consequently events are also versioned. There is one line per event and the version and the cancel date (M_CNL_DATE) have been updated following the action. This new behavior introduces the notion of action on events (field M_ACTION in TRN_EXT_DBF). Having the label of the event is not enough to understand what has been performed, the action like insert, cancel or cancel and re−event should be specified.

3.2.1.5 − Market operation on linked trade

A same market operation can impact several trades for example a netting. But in case of a linked trade, one market operation will be created for each trade in Mx G2000. In Mx.3, when an event is performed at the level of the contract or the package, only one event reference will be created.

3.2.1.6 – Settlement flows migration

Initially in Mx G2000, it was possible to have only one settlement flow when inserting a market operation. This unique flow was stored in the table MKT_OP_DBF. This table contains one row per market operation that's why settlement flows can be stored in the same table.

After enhancements, Mx G2000 allows users to enter several settlement conditions for one market operation. Consequently, fields about settlements in the table MKT_OP_DBF were not filled anymore. Those settlements flows were stored in the table TRN_HDRF_DBF that can be linked to other market operation tables using a 1−N join. To retrieve in this table the settlement flows associated to market operations, it is necessary to filter on rows where M_FLOW_FMLY = "SPB_MOP" or (M_FLOW_FMLY = "SPB_TRNRPL" and MOP_NB <> 0) because this table contains the settlement flows but also additional flows and past flows related to restructured deals.

In Mx.3, it is possible to enter several flows for the same event. The table of the settlement flows is called FCE_FLOW_DBF and can be linked to the table EVT_EVENT_DBF with a 1−N join. Each record representing a settlement flow in the table FCE_FLOW_DBF has two columns to trace the history of the flow:

- M_VERSION: version of the trade when the settlement flow was created.
- M_CVERSION: version of the trade when the settlement flow is no longer available.

For example, if we apply an unwind on a trade version =3 and we insert a settlement flow, this generates the version 4 of the trade and inserts a new record for the settlement flow in the table FCE_FLOW_DBF where :

- M_VERSION: is equal to 4
- M_CVERSION: is equal to 0

Other events occur on the trade and now the trade version is 6 for instance. We apply a Cancel of the previous unwind. The trade version is now 7 and a the settlement flow in the table FCE_FLOW_DBF has now:

- M_VERSION: is still equal to 4
- M_CVERSION: is equal to 7

Therefore, to retrieve "live" settlement flows for a deal, it is possible to do directly a join between TRN_HDR_DBF and FCE_FLOW_DBF adding the condition M_CVERSION = 0.

Consequently, flows originally stored in one table TRN_HDRF_DBF are in Mx.3 stored in two tables TRN_HDRF_DBF and FCE_FLOW_DBF that are not linked together.

Migration implementation choice: As no SQL join exists between those tables, a view TRN_HDRF_VW_DBF is provided by Murex to bring together records coming from TRN_HDRF_DBF and FCE_FLOW_DBF thanks to a SQL union.

The corresponding SQL is describe in the appendix "Additional flows view".

3.2.2 – Fields Mapping

3.2.2.1 – Migration implementation choice

Views are provided by Murex to hide cardinality differences and naming convention differences concerning the tables TRN_EXT_DBF, EVT_EVENT_DBF and FCE_FLOW_DBF.

The corresponding SQL is described in the appendix "Market operation views".

Generally, the table TRMKTOP_DBF is not referenced in Mreports because it requires 1−n joins which could not be originally implemented. Reports are commonly built with a 1−1 relation between TRN_HDR_DBF and MKT_OP_DBF based on the field M_OPT_MOPNB. This relation was mostly used to retrieve the last market operation. Its equivalence is provided by the views that retrieves one event per trade, more specifically the last impact event. To retrieve all market operations, the views should not be used.

In Mx G2000, the cancel of a market operation was not recorded in the tables like their insertion. Instead, it provoked the deletion of the corresponding market operation from the table. The view reproduces this behavior by just returning the insertions and the re−events (whenever cancel and re−events are performed).

As historically settlements can be unique or multiple, three views are provided. Following client needs, one of them can be used for the mapping:

- MKT_OP_VW_DBF: view without any settlement
- MKT_OP_ONES_VW_DBF: view with one settlement, the settlement has been chosen arbitrarily with the maximum reference
- MKT_OP_ALLS_VW_DBF: view with all settlements.

Note that to allow using compatibility views, the default command /DD_COMPATIBILITY in the launcherall.mxres file is mandatory.

It is not recommended to use views automatically. Although views do facilitate the migration of the report, they introduce performance problems and complexity to the report.

3.2.2.2 – Data model impact

- For market operations:

2.10 Table	2.10 Field	Description	3.1 Table	3.1 Field	3.1 Mapping using the DB view	Commen
MKT_OP	M_NB	Internal Mop number	TRN_EXT EVT_EVENT	M_EVT_REF M_REFERENCE	MKT_OP_VW_ DBF.M_NB MKT_OP_ONES _VW_DBF.M_NB MKT_OP_ALLS _VW_DBF.M_NB	Event num
MKT_OP	M_AGREED_ OP	Validation step	Not maintained	Not maintained		
MKT_OP	M_AREA_ CODE	Area code	EVT_EVENT	M_AREA_CODE		
KT_OP	M_BO_CMT	Bo Commitment	Not maintained	Not maintained		
MKT_OP	M_BO_CNF	Bo Confirmation	Not maintained	Not maintained		
MKT_OP	M_BO_SGN	Bo signature	Not maintained	Not maintained		
MKT_OP	M_COMMENT	Mop comment	EVT_EVENT	M_COMMENT	MKT_OP_VW_ DBF.M_COMMENT MKT_OP_ONES_VW_ DBF.M_COMMENT MKT_OP_ALLS_VW_ DBF.M_COMMENT	Event com
MKT_OP	M_DATE	Mop date	TRN_EXT	M_DATE	MKT_OP_VW_ DBF.M_ DATE MKT_OP_ONES_VW_ DBF.M_DATE MKT_OP_ALLS_VW_	Event date insertion. I returns the insertion c retrieve th

					DBF.M_DATE	acceptanc use M_ACT_D
MKT_OP	M_DEST	Settlement Destination (portfolio/counterpart)	FCE_FLOW	M_DEST	MKT_OP_ONES_VW_ DBF.M_DEST MKT_OP_ALLS_VW_ DBF.M_DEST	M_DEST i reference. internal de (M_INTER a join with TRN_PFL needed to the portfol else for ex deals (M_INTER 0), use a j TRN_CPD
MKT_OP	M_DEST_NB	Destination trade number	TRN_EXT	M_TRADE_REF	MKT_OP_VW_ DBF.M_TRADE_REF MKT_OP_ONES_ VW.M_TRADE_REF MKT_OP_ALLS_ VW.M_TRADE_REF	Trade num
MKT_OP	M_HDG_SOLD	Hedge transaction sold	TABLE#DATA#EVENT TABLE#DATA#DEAL...	M_HDG_SOLD		Hedge tra sold
MKT_OP	M_INSTRUMENT	Instrument	TRN_HDR	M_INSTRUMENT		
MKT_OP	M_LAT		TRN_EXT	M_LAT	MKT_OP_VW_ DBF.M_ LAT MKT_OP_ONES_ VW_ DBF.M_LAT MKT_OP_ALLS_ VW_ DBF.M_LAT	Event Acc flag
MKT_OP	M_MKT_INDEX		TRN_HDR	M_MKT_INDEX		
MKT_OP	M_ORIGIN	Settlement (Source portfolio label)	FCE_FLOW	M_SOURCE	MKT_OP_ONES_ VW_DBF.M_ORIGIN MKT_OP_ALLS_ VW_DBF.M_ORIGIN	M_SOURC reference. retrieve th portfolio la join with TRN_PFL needed.
MKT_OP	M_ORIGIN_NB	Original trade number	TRN_HDR	M_CREATOR		
MKT_OP	M_PL_INSCUR	Instrument p&l currency	TRN_HDR	M_PL_INSCUR		
MKT_OP	M_PL_KEY1	P&L key	TRN_HDR	M_PL_KEY1		
MKT_OP	M_PURGE_DATE	Purge date	Not maintained	Not maintained		
MKT_OP	M_PURGE_GRP	Purge group	Not maintained	Not maintained		

MKT_OP	M_PURGE_STS	Purge section	Not maintained	Not maintained		
MKT_OP	M_RSKSECTION	Risk Section	TRN_HDR	M_RSKSECTION		
MKT_OP	M_SYS_DATE	System date	TRN_EXT	M_LOGIN_DATE	System date	
New	New	New	TRN_EXT	M__DT_TS	Timestamp date	
MKT_OP	M_TIME	Mop time	TRN_EXT	M__DT_TS	Timestamp time	
MKT_OP	M_TIME_ZONE	Time zone	TRN_HDR	M_TIME_ZONE		This field s in TRN_H but this no not used a as the trac directly sa the correc
MKT_OP	M_TIME_ZONEA		Not maintained	Not maintained		
MKT_OP	M_TRADER	Trader label	EVT_EVENT	M_USER	MKT_OP_VW_ DBF.M_TRADER MKT_OP_ONES_ VW_DBF.M_TRADER MKT_OP_ALLS_ VW_DBF.M_TRADER	
MKT_OP	M_TRN_FMLY	Transaction family	TRN_HDR	M_TRN_FMLY		
MKT_OP	M_TRN_GRP	Transaction group	TRN_HDR	M_TRN_GRP		
MKT_OP	M_TRN_TYPE	Transaction type	TRN_HDR	M_TRN_TYPE		
MKT_OP	M_TYPE	Mop typology	CLASS_MAPPING	M_NAME	MKT_OP_VW_ DBF.M_TYPE	Retrieved event typo reference
MKT_OP	M_TYPE_SUB	Mop sub−typology	Not maintained	Not maintained	MKT_OP_VW_ DBF. M_SUB_TYPE	Deprecate type is nov field
MKT_OP	M_VAL_DATE	Validation date	In the work flow space	In the work flow space		
MKT_OP	M_VAL_STATUS	Validation status	In the work flow space	In the work flow space		

- For TRN_HDR_DBF

2.10 Table	2.10 Field	Description	3.1 Table	3.1 Fields	Comment
TRN_HDR	M_MOP_CREAT	Mop creator	TRN_DD	M_CEVTCL	Class id of event creator (can be exercise, restructure or C&R)
TRN_HDR	M_MOP LAST	Last Mop reference	TRN_DD	M_LIMPECL	Event Class id of last impact. This last impact is the impact that has not been canceled and with the most recent date.

TRN_HDR	M_OPT_MOPLST	Last Mop value date	TRN_DD	M_LIMPDT	Impact date of the impact
TRN_HDR	M_OPT_MOPLSD	Last mop system date	TRN_DD	M_LIMPSDT	System date of last impact
TRN_HDR	M_OPT_MOPNB	Mop number	Deprecated	Deprecated	
TRN_HDR	M_OPT_MOPFST	First Mop Date	Not maintained	Not maintained	Request to retrieve equivalent information: select (select ext1.M_DATE from TRN_EXT_DBF ext1 where ext1.M_ORIG_TRADE = ext.M_ORIG_TRADE and ext1.M_VERSION = (ext.M_VERSION +1)) as M_OPT_MOPFST from TRN_EXT_DBF ext group by M_TRADE_REF having ext.M_VERSION = min (M_VERSION)
TRN_HDR	M_OPT_MOPSUB	Mop subtype no more in Mx.3	Not maintained	Not maintained	
TRN_HDR	M_MOP_CRSUB	CMM conversion	Not maintained	Not maintained	
None	None		TRN_DD	M_LIMPVS	Last impact version
None	None		TRN_DD	M_LIMPEXTREF	Reference to TRN_EXT of last impact insertion
None	None		TRN_DD	M_CVS	vers
			TRN_DD	M_LEXTUDFREF	Reference to retrieve the last version of the UDFs, to link with UDF tables.
None	None		TRN_DD	M_LIMPREF	Reference link with the table EVT_IMP_DBF to retrieve last impact information

Note that the fixing is not considered as an impact and consequently will be not returned by the last impact fields.

- For Settlement conditions

2.10 Table	2.10 Field	3.1 Table	3.1 Fields	3.1 Mapping using the DB view	Comment
MKT_OP TRN_HDRF	M_CURRENCY M_CURRENCY	FCE_FLOW	M_CURRENCY	MKT_OP_ONES_VW_DBF.M_CURRENCY MKT_OP_ALLS_VW_DBF.M_CURRENCY	Settlement currency
MKT_OP TRN_HDRF	M_NFAMOUNT M_AMOUNT (signed)	TRN_HDR and FCE_FLOW	M_AMOUNT (unsigned).	MKT_OP_ONES_VW_DBF.M_NFAMOUNT MKT_OP_ALLS_VW_DBF.M_NFAMOUNT (unsigned)	To retrieve signed amount; you need to join with TRN_HDR: when T.M_BINTERNAL = Y; and T.M_SINTERNAL = Y; and T.M_COMMENT_BS

					= S; then F.M_AMOUNT * (–1) else F.M_AMOUNT end)
MKT_OP TRN_HDRF	M_NFAMOUNTD M_DATE	FCE_FLOW	M_VAL_DATE	MKT_OP_ONES_VW_DBF.M_NFAMOUNTD MKT_OP_ALLS_VW_DBF.M_NFAMOUNTD	Settlement date
TRN_HDRF	M_COMMENT	FCE_FLOW	M_COMMENT	MKT_OP_VW_DBF.M_COMMENT	Settlement comment
MKT_OP	M_S_PRICE	Deprecated	Deprecated		
None	None	FCE_FLOW	M_DATE		Event Date

3.2.2.3 – Dynamic table impact

In Mx G2000, information about market operations in dynamic tables are provided by fields starting with TP_MOP. Also market operations are carried by the original trade. For example in case of a cancel and reissue, the TP_MOPLST field return the C&R information at the level of the canceled trade (original trade).

In Mx.3, the TP_MOP fields should not be used, they have been replaced by several new sets of fields.

–set of fields describing the origin event, that start with CNT_EVT

–set of fields describing the last event, that start with CNTLEVT

–set of fields describing the event that turned the deal into the "Not available " status, that start with CNTNEVT.

–set of fields describing the last event that impacts the deal, that start with CNTLIMP

Events are now carried by the issued trade because events are stored at the level of the trade version. For example, after a cancel and reissue, the original trade becomes not available in version 1 and the second trade is created in version 2. Information about the C&R is returned by the issued trade using the set of fields CNTLEVT. However, if we need to have the information at the level of the original trade (specially for migration purpose), the set of fields CNTNEVT can be used as they return the event that is the cause of the not available status.

The last event set of fields CNTLEVT returns information about the last event that happen to the deal. It can be an event insertion, a cancel or a cancel and reissue (note that fixings are not considered as an impact here). To retrieve the last impact of the trade (if it is not necessary to consider the canceled event, but the last effective event), the set of fields last impact CNTLIMP can be used.

To facilitate the join between dynamic tables and the table TRN_EXT_DBF, the reference of the available extension for a each trade is given by the following fields: CMP_EXT in TRNRP_PL and V_CMP_EXT in TRNRP_MK.

V2.1 field	V.2.10 description	V3.1field	V3.1description
TP_MOPCRT	Origin market operation	CNTCEVTCL	Contract creator event class id
TP_MOPCRTD	Market operation first date	CNT_EVTDT	Origin event insertion date. To retrieve the P&L acceptance date, use CNT_EVTAD
TP_MOPCRTL	Origin market operation (label)	Deprecated	Join with CLASS_MAPPING_DBF.M_NAME
TP_MOPCRTS	Origin market operation	Deprecated	Deprecated

	(sub−type)		
None	None	CNTLEVTCL	Contract last event class id
None	None	CNTLEVTDT	Contract last event insertion date. To retrieve the P&L acceptance date, use CNTLEVTAD
None	None	CNTLEVT	Contract last event reference
None	None	CNTLEVTAC	Contract last event action
None	None	CNTLEVTAD	Contract last event actual date
None	None	CNTLEVTOR	Contract last event origin reference
None	None	CNTLEVTVS	Contract last event version
None	None	CNTLVTTSD	Contract last event timestamp date
None	None	CNTLVTTST	Contract last event timestamp time (ms)
None	None	CNT_EVT	Contract origin event reference
None	None	CNT_EVTADT	Contract origin event actual date
None	None	CNT_EVTOR	Contract origin event origin reference
None	None	CNT_EVTTSD	Contract origin event timestamp date
None	None	CNT_EVTTST	Contract origin event timestamp time (ms)
None	None	CNT_EVTVS	Contract origin event version
None	None	CNTNAEVT	Contract Not Available event reference
None	None	CNTNEVTAC	Contract Not Available event action
None	None	CNTNEVTAD	Contract Not Available event actual date
None	None	CNTNEVTCL	Contract Not Available event class id
None	None	CNTNEVTDT	Contract Not Available event date
None	None	CNTNEVTOR	Contract Not Available event origin reference
None	None	CNTNEVTVS	Contract Not Available event version
None	None	CNTNVTTSD	Contract Not Available event timestamp date
None	None	CNTNVTTSH	Contract Not Available event timestamp time
None	None	CNTNVTTST	Contract Not Available event timestamp time (ms)
None	None	CNTLIMEVT	Contract last impact event reference
TP_MOPLSTD	Market operation: last date	CNTLIMPDT	Contract last impact event date
TP_MOPLST	Last market operation	CNTLIMPCL	Contract last impact event class id
TP_MOPLSTL	Last market operation (label)	M_NAME	Join CNTLIMPCL with CLASS_MAPPING_DBF.M_ID and return M_NAME
None	None	CNTLIMTSD	Contract last impact event timestamp date
None	None	CNTLIMTST	Contract last impact event timestamp time (ms)
None	None	CNTLIMTSH	Contract last impact event timestamp time
None	None	CNTLIMEXT	Contract last impact event extension
None	None	CMP_EXT in TRNRP_PL V_CMP_EXT in TRNRP_MK	Component extension reference

3.3 – Market operation Dynamic table TRNRP_MK

3.3.1 – Functionality

This dynamic table returns all market operations.

In Mx G2000, this table returns all trades and all market operations (one row per trade and per market operation). The PL fields represent the PL impact from the market operations (cash proceeds). Note that in Mx.3, these PL fields are not calculated. This will be a new development.

- Settlement condition

After enhancements in Mx G2000 allowing several settlements, the dynamic table TRNRP_MK still returning only one flow (the first one). In Mx.3, by default the dynamic table returns one line per trade per version. The creation code ACTIVATE_MULTIPLE_SETTLEMENTS gives one line per flow to display all settlement flows.

3.3.2 – Fields Mapping

V2.1 field	V.2.10 description	V3.1field	V3.1description
MK_NB	MktOpnumber	V_EVT_REF	Event reference
MK_TYPE	MktOptype(EXR,NET,..)	V_EVT_CLS	Event class label
MK_SUBTYPE	MktOpsubtype(CLOSE OUT)	Not maintained	
MK_DTE	MktOpdate	V_DATE	Version date
MK_DTEACC	MktOpsystemdate	V_TSD	Version system date (timestamp)
MK_TIME	MktOptime	V_TSH	Version system time (timestamp)
None	None	V_TST	Version timestamp time (ms)
MK_LEVEL	MktOplevel(2,3)		
MK_BOSGN	MktOpBOsignature	Not maintained	new fields stp status from the workflow
MK_BOCNF	MktOpBOconfirmation	Not maintained	new fields stp status from the workflow
MK_BOCMT	MktOpBOcommitor	Not maintained	new fields stp status from the workflow
None	None	V_STP_STS	STP validation level status
None	None	V_STP_STS0	STP status 0
None	None	V_STP_STS1	STP status 1
None	None	V_STP_STS2	STP status 2
None	None	V_STP_STS3	STP status 3
None	None	V_STP_STS4	STP status 4
MK_ORIGIN	MktOporiginaccount	V_CNT_SOURCE	Contractsource
MK_DEST	MktOpdestinationaccount	V_CNT_DEST	Contractdestination
MK_STLAMT	MktOpsettlementamount	V_EVTSTLAMT	Amountofcurrentsettlementflow
MK_ STLDTE	MktOpsettlementdate	V_EVTSTLDTE	Dateofcurrentsettlementflow
MK_ STLCUR	MktOpsettlementcurrency	V_EVTSTLCUR	Currencyofcurrentsettlementflow
None	None	V_EVSTLIND	Index of current event settlement flow
None	None	V_EVSTLNB	Number of event settlement flows
MK_ STLPRC	MktOpsettlementprice	Not maintained	
MK_ QTY	MktOpquantity	V_IMPQTY	Current impacted quantity
None	None	TRD_AQTY	Trade available quantity
None	None	V_AVQTY	Current available quantity
MK_ BROTHER	MktOpbrother	Not maintained	

None	None	V_ACTDT	Version actual date
None	None	V_ACTION	Version action
None	None	V_ACTOR	Version actor
None	None	V_EVT_OREF	Event origin reference
None	None	V_EVT_VER	Event version
None	None	V_LAT	Version acceptance flag
None	None	V_TRD_REF	Physical trade reference
None	None	V_VERSION	Current contract version

3.4 – Fixing

In Mx G2000, it is possible to fix using the fixing procedure. The configuration of the fixing procedure can be set from a housekeeper session and is stored in the table FXNG_CONF_DBF. Fixings are stored in the table FXNG_DBF. The fixing is not considered as a market event.

In Mx.3, the fixing can be performed using the fixing procedure or as an event at the level of the trade (fixing) which can be applied to individual trade without running the fixing procedure. The fixing is now considered as an event. Consequently, after a fixing, the trade is versioned in the table TRN_EXT_DBF and the event is stored in the table EVT_EVENT_DBF. Note that fixings are still being stored as well in the table FXNG_DBF and they can be linked to the event table thanks to the field M_EVT_REF which is filled with the event reference. Note that in Mx.3, there is a new information available in FXNG_DBF: the fixing date stored in the field M_FIX_DATE.

Deals coming from the Mx G2000 will be migrated with their records in FXNG_DBF table but they will not have any records in EVT_EVENT_DBF. The field M_OBS_FXNG in FXNG_DBF allows to distinguish deals that have already been fixed without any row in EVT_EVENT (M_OBS_FXNG=1).

The dynamic table TRNRP_XG still has in Mx.3 the same behavior it had in Mx G2000.

The audit of the fixing is stored in the following tables: AUD_FIXH_DBF and AUD_FIXB_DBF. The fixing audit table encompasses all audit trails of trades or indexes which have undergone the fixing procedure. The fixing audit table, however, does not capture the fixing event performed at the individual deal level via Operations>Fixing. The fixing event performed at the individual deal level will be audited instead at the trade audit level.

4 – Deliverables and Settlement instructions

4.1 – Deliverables description

4.1.1 – Data model schema

4.1.1.1 – In Mx G2000

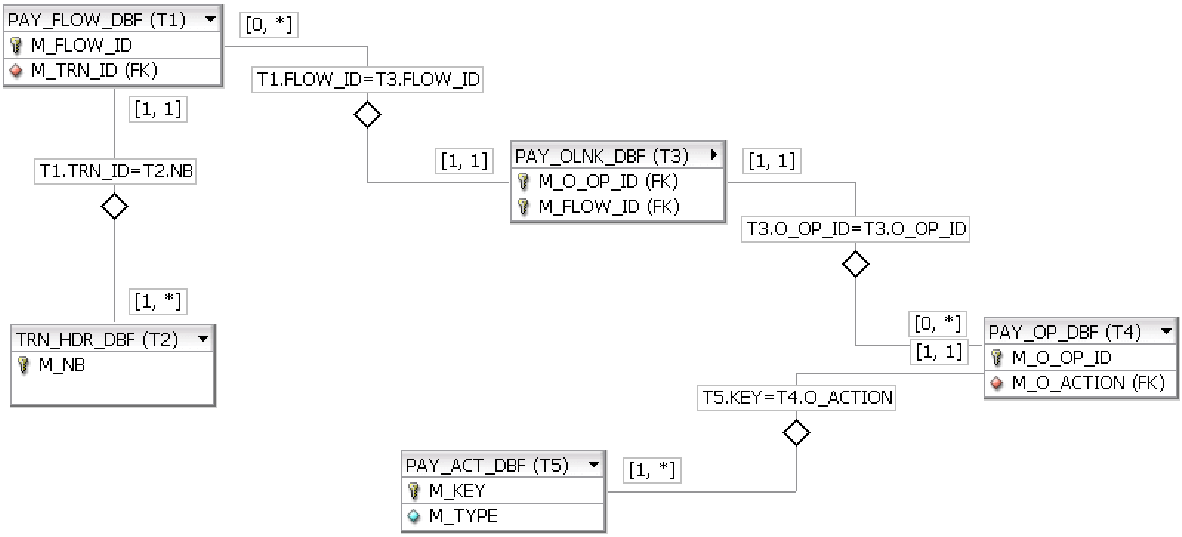

Figure 13: Payment module data model (v2.10)

4.1.1.2 – In Mx.3

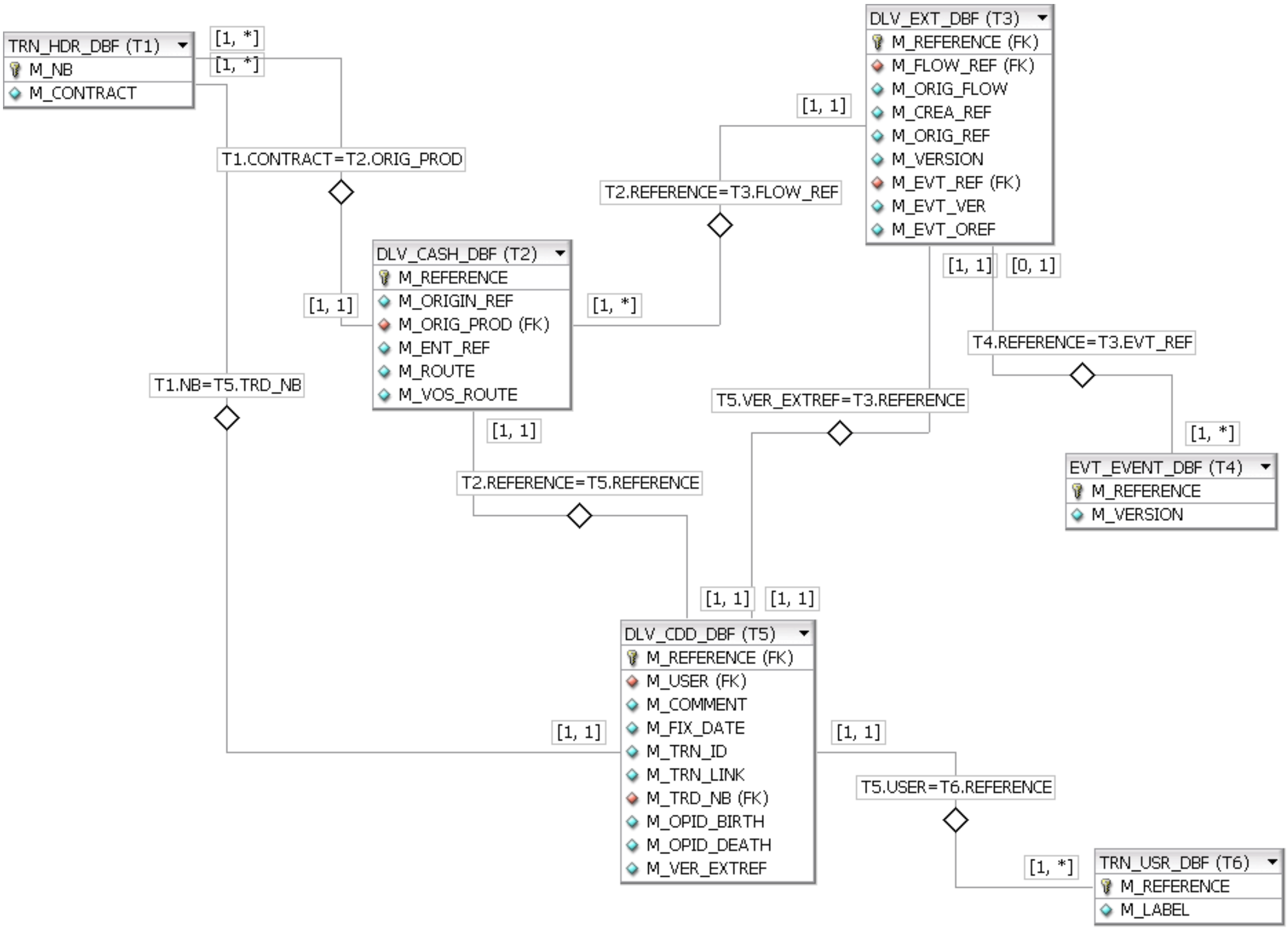

Figure 14: Deliverable module data model (v3.1)

4.1.2 – Description

Payments can be generated:

- automatically when inserting a trade: known flows, which payment date is included in the pay fix window, will generate payments (in the original status, for e.g.: M_STATUS="INIT")
- by a fixing
- by the pay fix refresh

4.1.2.1 – In Mx G2000

4.1.2.1.1 – Payments

Payments are stored in the table PAY_FLOW_DBF and are linked to the trade number. Market operations, actions (like validation actions), or operations (like netting, unetting, cancel, split) can be performed on a trade and will impact the flows.

4.1.2.1.2 – Market operations

Market operations are stored in the table MKT_OP_DBF. When performing a mop on the trade, flows can be impacted: if the generated flow is identical to the one it replaces, nothing happens otherwise, the original flow is canceled by a cancel operation (PAY_OP_DBF.M_O_ACTION = @CNL and PAY_FLOW_DBF.M_STATUS to "CNCL") and a new flow is inserted with a new reference.

The remove mop acts as an other operation: original and resulted flows will be compared: new flows will be inserted and modified flows will be canceled and replaced.

4.1.2.1.3 – Operations and actions

Actions are stored in the tables PAY_ACT_DBF and PAY_AIO_DBF. Operations are stored in the table PAY_OP_DBF and linked to the flows thanks to the table PAY_OLNK_DBF.

Operations are stored in the tables PAY_OP_DBF and PAY_OLNK_DBF. Here are the possible values for operation (M_O_ACTION in PAY_OP_DBF):

- @NET: Netting. Several flows become one flow. Original flows and resulted flow are linked to the operation. For original flows, M_STATUS= "CNL" and for the resulted flow M_STATUS = "NET". Note that the trade reference field TRN_REF is empty for the resulted flow.
- @UNT: Unetting. Original flows and resulted flow are linked to the operation. For original flow, M_STATUS= initial status and for the resulted flow M_STATUS = "CNL".
- @MOD: Modification of the settlement instruction.
- @CNL: Cancel of the flow (consequence of a market operation)
- @SPL: Split. One flow becomes several flows. Original flow and resulted flows are linked to the operation.
- @INV: Netting with a specific netting key. Same behavior as netting.

When performing an action or an operation on the trade, a new line is inserted in PAY_OP_DBF and the action/operation is stored in the field M_O_ACTION. Each action and operation will insert a new line in PAY_OP_DBF.

Note that when an already sent flow needs to be modified, for example after a cancel and reissue of the trade, this flow will be reversed with a reversal flow, then the new flow will be inserted. This mechanism still the same in Mx.3 and a new field has been created M_REV_STS returning 1 if the flow is reversal (canceling flow), 2 if the flow is reversed (canceled flow) else 0.

4.1.2.1.4 – Netting

For the netted flows, the field M_OPID_DEATH is filled by the id of the netting operation. Whereas for the resulted flows, this information is stored in the field M_OPID_BIRTH/.

Consequently,

- To retrieve the netted flows: M_OPID_DEATH <> 0
- To retrieve a netting result: M_OPID_BIRTH <> 0
- To retrieve the netted flows from the resulted flows, it is possible to use the following query:

```
select M_FLOW_ID from PAY_FLOW_DBF
where M_OPID_DEATH = 'M_OPID_BIRTH'
```

4.1.2.2 – In Mx.3

4.1.2.2.1 – Deliverable cash

Payments are considered as deliverable and managed by the deliverable module. Note that in the deliverable module there is also deliverable physical, deliverable security... but payments concerns only deliverable cash.

Deliverable cash are stored in the table DLV_CASH_DBF and tracable through the origin contract number (field M_ORIG_PROD). In DLV_CASH_DBF, there are two fields, M_PRODUCER and M_ORIG_PROD, containing the original contract number: M_PRODUCER will be empty when performing an event at the level of the flow whereas the field M_ORIG_PROD remains filled except after a netting of flows coming from different origin contract number.

If an event is performed at the level of the trade, DLV_CASH_DBF.M_PROD_CVER enables to retrieve the version of the producer that canceled the flow. But this field, along with other fields linked to the producer, are not filled if the event is done at the level of the flow.

In Mx.3, a new table DLV_CDD_DBF has been created as an extension of the table DLV_CASH_DBF to allow compatibility and to facilitate the migration. The join between this table and DLV_CASH_DBF is a 1−1 join on the flow number.

Note that to allow the use of this compatible table DLV_CDD_DBF by MX, it is necessary to add the default command /DD_COMPATIBILITY in the launcherall.mxres file.

4.1.2.2.2 – Validation Actions

Actions are migrated in the workflow. This part is out of scope of this document.

Some validation actions can be defined to change the status of the trade to cancel and consequently cancel the flows. Those actions are migrated as cancel events. The validation actions which will be migrated as cancel events are the actions in PAY_ACT_DBF where M_DST_STATUS = (select M_STS_CNCL from PAY_CFG_DBF).

4.1.2.2.3 – Events

Market operations and flow operations are migrated as events. In Mx.3,it is possible to act directly at the level of the flow.

Remove mop and unetting are migrated as a cancel of event.

Here is the mapping:

Actions/Operations in 2,10	Description	Class id of the event in MX.3	Action of the event in MX.3	Description
@NET	Netting	MxVbi59185	1 (insert)	Netting event
@UNT	Unetting	MxVbi59185	2(delete)	Cancel of a netting event
@MOD	Settlement instruction modification	Not maintained	Not maintained	No equivalence in MX.3
@CNL	Cancel	Not maintained	Not maintained	This was not an operation so no event equivalence. But a new version of the flow is stored in DLV_EXT and the field M_PROD_CVER is filled to indicate that the flow has been canceled
@SPL	Split	MHoJu69321	1 (insert)	Split event
@INV	Specific netting	MxVbi59185	1 (insert)	Netting event
Action of cancel	Following PAY_CFG.M_STS_CNCL	MwXJC40268	1 (insert)	Cancel event

In Mx.3, operations are migrated as events. Each time an event is performed, flows can be impacted and versioned in the table DLV_EXT_DBF. Each time an event is performed on flows original flows and new flows are compared and only different flows are versioned. If a new flow is inserted, its version will be equal to 1. In the table DLV_EXT_DBF, there is one record per flow per version. This leads to a 1–N join between DLV_CASH_DBF and DLV_EXT_DBF. To retrieve available flows, it is necessary to look for the last version of the flow for each original flow id: max(M_VERSION) group by M_ORIG_FLOW in DLV_EXT_DBF.

The DLV_CDD_DBF table facilitates the join with the table DLV_EXT_DBF as it allow a 1–1 join with the last version of each flow reference.

Event information can be retrieved for each version in the table EVT_EVENT_DBF (as well as events performed on trades). Note that event information are hold by original and resulted flows.

4.1.2.2.4 – Netting

As original flows and resulted flows hold the event, it is easy to find the link between contributing flows and resulted flows. The field M_EVT_INTID (DLV_EXT_DBF.MEVT_INTID='MxVbi58185') will contain the netting class id and the field M_EVT_REF will contain the id of the netting. Also to distinct between these flows and the resulted flow, as it is a new flow, it is possible to use the field M_VERSION=1.

To distinct intra–deal netting from inter–deal netting, it is possible to use the field M_ORIG_PROD as this field is empty when the netting is inter–deal.

4.1.3 – Fields mapping

4.1.3.1 – Migration implementation choice

Depending on reports, it is possible to migrate them either using new Mx.3 datamodel or using the deliverable dynamic table.

A new Mx.3 table DLV_CDD_DBF is provided by Mx to facilitate the compatibility. This table provides compatibility fields and thanks to the field M_VER_EXTREF allows a 1–1 join between DLV_CASH_DBF, DLV_EXT_DBF, DLV_CDD_DBF in order to bypass 1–n cardinalities which are not supported by Mreport.

Note that to use the compatibility tables, the default command /DD_COMPATIBILITY in the launcherall.mxres file is mandatory.

4.1.3.2 – Flow tables

4.1.3.2.1 – PAY_FLOW_DBF table

The payment table PAY_FLOW_DBF

2.10 Table	2.10 Field	Comment	3.1 Table	3.1 Fields	Comment	Mapping with Mx.3 dynamic table fields
PAY_FLOW	M_INST_TYPE	Instrument type 0:Cash 1:delivery		Not maintained in Mx.3	Not maintained as DLV_CASH only contains type 0:cash and DLV_PHYS contains type 1:delivery	
PAY_FLOW	M_FLOW_ID	Cash flow id	DLV_CASH	M_REFERENCE	Cash flow reference	DF_REF
PAY_FLOW	M_OPID_DEATH	Operation id for dead cash flow (join PAY_OP)	DLV_CDD	M_OPID_DEATH	Event id that cancels the flow	DC_OIDEATH
PAY_FLOW	M_OPID_BIRTH	Operation id for new cash flow (join PAY_OP)	DLV_CDD	M_OPID_BIRTH	Event id that creates the flow	DC_OIBIRTH
PAY_FLOW	M_USER	User name	DLV_CDD	M_USER	User name	DC_USER
PAY_FLOW	M_ACTION	Action		In the workflow		
PAY_FLOW	M_SYS_DATE	System date	DLV_EXT	M_LOGIN_DATE		
PAY_FLOW	M_CPU_DATE	Computer date	DLV_EXT	M__DT_TS	For migrated deliverables: insertion date of the trade version; For new deliverables: insertion date of the deliverable	DF_CMPDATE
PAY_FLOW	M_CPU_TIME	Computer time		Not maintained in Mx.3	This information can be deduced from M__DT_TS	
PAY_FLOW	M_TRN_ID	Trade number that generates the flow	DLV_CDD	M_TRN_ID	Trade number that generates the flow	DC_TRNID
PAY_FLOW	M_MOP_ID	Mop number	DLV_EXT	M_EVT_REF	In Mx.3, trade event and deliverable event are in the same table EVT_EVENT. Always filled in Mx.3.	DF_EVTREF
PAY_FLOW	M_TRN_FMLY	Transaction family	TRN_HDR	M_TRN_FMLY	Transaction family	DF_TFAMILY
	M_TRN_GRP		TRN_HDR	M_TRN_GRP	Transaction group	DF_TGROUP

PAY_FLOW		Transaction group				
PAY_FLOW	M_TRN_TYPE	Transaction type	TRN_HDR	M_TRN_TYPE	Transaction type	DF_TTYPE
PAY_FLOW	M_ENTITY	Entity	TRN_ENTD or TRN_CPDF	M_LABEL M_DSP_LABEL	Following client choices: Id of the Closing Entity. Join with the closing entity table TRN_ENTD DLV_CASH.M_ENT_REF = TRN_ENTD.M_REF or Id of the Legal Entity. Join with the counterparty table TRN_CPDF DLV_CASH.M_LENT_REF = TRN_CPDF.M_ID At flow level closing entity = legal entity except for internal deals which don't produce any payments	DF_ENT or DF_LENT
PAY_FLOW	M_PHASE	Phase	DLV_CASH	M_PHASE	Phase	DF_PHASE
PAY_FLOW	M_LEG	Leg	DLV_CASH	M_LEG	Leg	DF_LEG
PAY_FLOW	M_CALC_DATE0	Calculation start date	DLV_CASH	M_P_START_DT	Period start date	DF_PSDATE
PAY_FLOW	M_CALC_DATE1	Calculation end date	DLV_CASH	M_P_END_DT	Pariod end date	DF_PEDATE
PAY_FLOW	M_FIX_DATE	Fixing date	DLV_DD	M_FIX_DATE	Under development	DC_FIXDATE
PAY_FLOW	M_VALUE_DATE	Value date	DLV_CASH	M_VAL_DATE	Value date	DF_VDATE
PAY_FLOW	M_REL_DATE	Release date	DLV_CASH	M_REL_DATE	Release date	DF_RDATE
PAY_FLOW	M_REL_TIME	Release time		Not maintained in Mx.3		
PAY_FLOW	M_E_STL_DATE	Securities – Settl. date	DLV_EXT	M_	Effective date	
PAY_FLOW	M_E_AMOUNT	Securities – Amount		Not maintained in Mx.3		
PAY_FLOW	M_CURRENCY	Payment currency	DLV_CASH	M_CURRENCY	Deliverable currency	DC_CUR
PAY_FLOW	M_FLOW_TYPE0	Cash flow type 0	FLOW_TYPO	M_TYPE0	Join DLV_CASH.M_TYPOLOGY = FLOW_TYPO.M_REF	DF_TYPE0
PAY_FLOW	M_FLOW_TYPE1	Cash flow type 1	FLOW_TYPO	M_TYPE1	Join DLV_CASH.M_TYPOLOGY = FLOW_TYPO.M_REF	DF_TYPE1
PAY_FLOW	M_FLOW_TYPE2	Cash flow type 2	FLOW_TYPO	M_TYPE2	Join DLV_CASH.M_TYPOLOGY = FLOW_TYPO.M_REF	DF_TYPE2
				M_TYPE3		DF_TYPE3

PAY_FLOW	M_FLOW_TYPE3	Cash flow type 3	FLOW_TYPO		Join DLV_CASH.M_TYPOLOGY = FLOW_TYPO.M_REF	
PAY_FLOW	M_FLOW_TYPE4	Cash flow type 4	FLOW_TYPO	M_TYPE4	Join DLV_CASH.M_TYPOLOGY = FLOW_TYPO.M_REF	DF_TYPE4
PAY_FLOW	M_MANUAL	Manual N=no Y=yes	DLV_CASH	M_MANUAL	Manual 0=no 1=yes	DF_MAN
PAY_FLOW	M_AMOUNT	Amount	DLV_CASH	M_QUANTITY	Deliverable amount	DF_QTY
PAY_FLOW	M_AMOUNT_RND	Rounded Amount	DLV_CASH	M_R_QUANTITY	Deliverable rounded amount	DC_RNDQTY
PAY_FLOW	M_CREDIT	C:credit D:debit	DLV_CASH	M_S_R	2=Receive 1=Sent If (M_S_R =2) Then 'C' else 'R'	DF_PR
PAY_FLOW	M_CNTRP	Counterpart	DLV_CASH	M_BENF_PARTY	Id of the counterpart. Join with the counterpart table to retrieve the label CASH.M_BENF_PARTY = CPDF.M_ID	DF_BPARTY
PAY_FLOW	M_NOSTRO_SI	Nostro default SI group nb Always filled whereas one only SI (default or specific) mode is used	DLV_CASH	M_ROUTE if R_SEL_MOD <> 64 else 0 M_ROUTE filled with the correct SI mode	Following M_R_SEL_MOD. See mapping below.	If DF_RSMN = 0 then DF_NSIREF else ''
PAY_FLOW	M_VOSTRO_SI	Vostro default SI group nb Always filled whereas one only SI (default or specific) mode is used	DLV_CASH	M_VOS_ROUTE if R_SEL_MOD <> 256 else 0 M_VOS_ROUTE filled with the correct SI mode	Following M_R_SEL_MOD. See mapping below.	If DF_RSMV = 0 then DF_VSIREF else ''
PAY_FLOW	M_NOSTRO_SIS	Nostro specific (flow or deal level) SI grp nr Always filled whereas one only SI (default or specific) mode is used	DLV_CASH	M_ROUTE if R_SEL_MOD = 64 else 0 M_ROUTE filled with the correct SI mode	Following M_R_SEL_MOD. See mapping below.	if DF_RSMN = 1 or 3 then DF_NSIREF else ''
PAY_FLOW	M_VOSTRO_SIS	Vostro specific (flow or deal level)	DLV_CASH	M_VOS_ROUTE if R_SEL_MOD = 256 else 0	Following M_R_SEL_MOD. See mapping below.	If DF_RSMV = 1 or 3 then DF_VSIREF

		SI grp nb Always filled whereas one only SI (default or specific) mode is used		M_VOS_ROUTE filled with the correct SI mode		else ''
PAY_FLOW	M_TRN_NETAGR	Transaction netting agreement	DLV_CDD	M_NETAGR	Transaction netting agreement	DF_TNETAG
PAY_FLOW	M_NETAGR	Payment netting agreement	TRN_CPDF	M_PAY_NET	Netting agreement of the flow counterpart	DF_NETAG
PAY_FLOW	M_STATUS	Cash flow status	DLV_CASH	M_STP_STATUS	Deliverable status	DF_STATUS
PAY_FLOW	M_COMMENT	Comment	DLV_CDD	M_COMMENT	Comment	DC_COMMENT
PAY_FLOW	M_TRN_LINK	Trade number else 0 when the flow should not be evaluated (flow reversed or replaced after a mop)	DLV_CDD	M_TRN_LINK	Trade number or 0 if the flow is cancelled or reversed. Note the he new field M_REV_STS (revaluation status) gives the following values: 0 : should be evaluated, 1: reversal, 2 : reversed. TRN_LINK can be recalculated with: case when M_REV_STS > 0 then 0 else M_TRD_REF end	DC_TRNLNK
PAY_FLOW	M_LTI_ID	Linked trade id		Not maintained in Mx.3		
PAY_FLOW	M_OD_AMNT	Amount in original denomination (CMM)		Not maintained in Mx.3		
PAY_FLOW	M_OD_AMNT_R	Rounded amount in original denomination (CMM)		Not maintained in Mx.3		
PAY_FLOW	M_OD_CUR	Original currency (CMM)		Not maintained in Mx.3		
PAY_FLOW	M_TYPOLOGY	Transaction typology	DLV_CASH	TRD_TYPO	Transaction typology	DF_TTYPO
PAY_FLOW	M_NET_SCAN	Netting scan. Resulting flow of a netting		Not maintained in Mx.3	The resulting flow can be identified with flow version = 1 and event class id is netting.	DF_ISSNET
PAY_FLOW	M_EVENT_ID	Flow event id. Always equal to 0.		Not maintained in Mx.3		

PAY_FLOW	M_SPE_N	Specific nostro (deal level) if specific deal =Y else N	DLV_CASH	M_R_SEL_MOD	See the mapping below	If DF_RSMN = 1 then 'Y' else 'N'
PAY_FLOW	M_SPE_V	Specific vostro (deal level) if specific deal =Y else N	DLV_CASH	M_R_SEL_MOD	See the mapping below	If DF_RSMV = 1 then 'Y' else 'N'
PAY_FLOW	M_SPE_NS	Specific nostro (flow level) if specific flow =Y else N	DLV_CASH	M_R_SEL_MOD	See the mapping below	If DF_RSMN = 3 then 'Y' else 'N'
PAY_FLOW	M_SPE_VS	Specific vostro (flow level) if specific flow =Y else N	DLV_CASH	M_R_SEL_MOD	See the mapping below	If DF_RSMV = 3 then 'Y' else 'N'
PAY_FLOW	M_TRN_REF	Transaction reference	DLV_CDD	M_TRN_REF	Trade number	DC_TRDNB
PAY_FLOW	M_NOS_DATE	Nostro date: when the flow is received in the balance for the first time		Not maintained in Mx.3	New Nostro management module	
PAY_FLOW	M_ACC_SECT	Accounting section	DLV_CASH	M_ACC_SECT	Accounting section	DF_ACCSEC
PAY_FLOW	M_TRD_SECT	Trading section (from portfolio)		Not maintained in Mx.3		
PAY_FLOW	M_VALD_DATE	Validity date	DLV_EXT	M_ACT_DATE	Actual date	DF_ADATE
PAY_FLOW	M_LAT_FLAG	Late trading		Not maintained in Mx.3		
PAY_FLOW	M_VOS_DATE	Vostro date		Not maintained in Mx.3		
None	None		DLV_CASH	PRODUCER	Origin contract number, empty if a mop occurred on the flow	
None	None		DLV_CASH	ORIG_PROD	Origin contract number, Always filled	
None	None		DLV_CASH	PROD_VS	Contract version	
None	None		DLV_CASH	REV_STS	Revaluation status. Return the following values: 0 : should be evaluated, 1: reversal, 2 : reversed.	
None	None		DLV_CASH	TRD_PARTY	If the flow is not manual = BENF_PARTY else 0	

None	None		DLV_CASH	TRD_REF	Origin trade number	DF_TRDREF

Comments on settlement instruction modes:

There are several types of SI mode:

- Default: use the default SI
- Specific deal: use SI modified at deal level
- Specific flow: use SI modified at flow level
- Customized deal (in Mx G2000.2.11 and Mx.3): use existing SI but not the default ones, changed manually at deal level
- Customized flow (n Mx G2000.2.11 and Mx.3): use existing SI but not the default ones, changed manually at flow level

In Mx.3, to retrieve the SI mode in the datamodel, the field M_R_SEL_MOD should be used. Here is the mapping for the field:

	M_R_SEL_MOD value	M_SPE_NS	M_SPE_N
NOSTRO_DEFAULT	1	0	0
NOSTRO_SPECIFIC_FLOW	64	1	0
NOSTRO_SPECIFIC	2	0	1
NOSTRO_CUSTOMIZED	4	0	0
NOSTRO_CUSTOMIZED_FLOW	128	0	0
	+		
	M_R_SEL_MOD value	M_SPE_VS	M_SPE_V
VOSTRO_DEFAULT	8	0	0
VOSTRO_SPECIFIC_FLOW	256	1	0
VOSTRO_SPECIFIC	16	0	1
VOSTRO_CUSTOMIZED	32	0	0
VOSTRO_CUSTOMIZED_FLOW	512	0	0

Or

	VOSTRO_DEFAULT	VOSTRO_SPECIFIC_FLOW	VOSTRO_SPECIFIC	VOSTRO_CUSTOMIZED	VOSTR FLOW
NOSTRO_DEFAULT	9	257	17	33	513
NOSTRO_SPECIFIC_FLOW	72	320	80	96	576
NOSTRO_SPECIFIC	10	258	18	34	514
NOSTRO_CUSTOMIZED	12	260	20	36	516
NOSTRO_CUSTOMIZED_FLOW	136	384	144	160	640

To retrieve the SI mode in the dynamic tables, the fields DF_RSMN and DF_RSMV should be used. Here is the mapping for the value of those fields:

0	Default

1	Specific deal
2	Customized deal
3	Specific flow
4	Customized flow

4.1.3.2.2 – Action tables

- PAY_ACT_DBF: Cash flow actions table
- PAY_AIO_DBF: List of templates messages for a given action

There is no equivalence in the data model for these tables in Mx.3 as all actions and operations are now in the workflow.

4.1.3.2.3 – Operation tables

- PAY_OLNK_DBF: Relation between PAY_FLOW and PAY_OP
- PAY_OP_DBF: List of operations table

4.2 – Settlement Instructions description

4.2.1 – Data model

- In Mx G2000

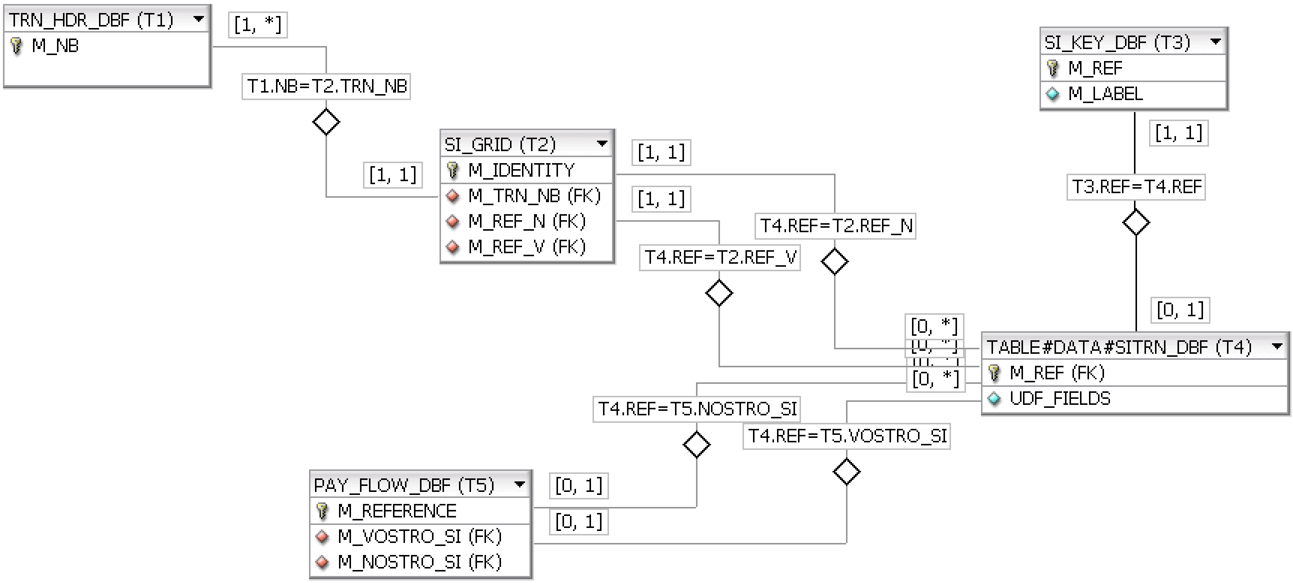

Figure 15: Settlement instructions data model (v2.10)

- In Mx.3

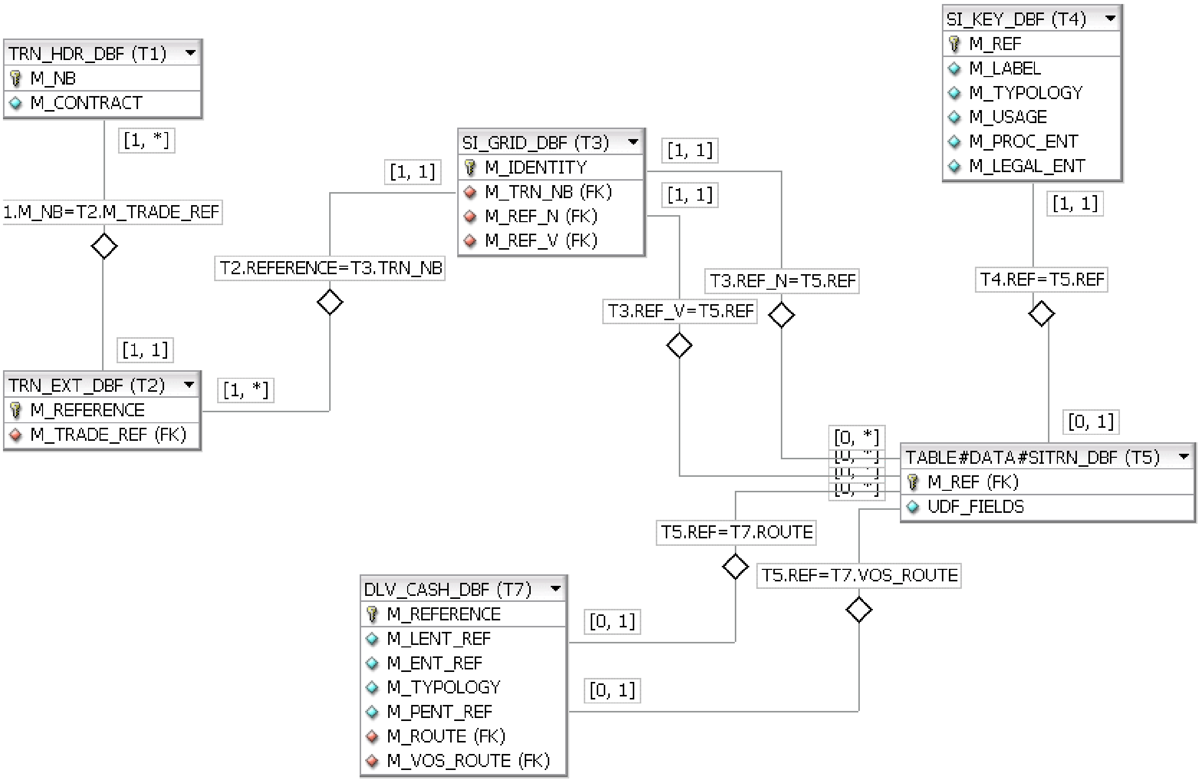

Figure 16: Settlement instructions data model (v3.1)

4.2.2 – Description

The settlement instructions are stored in the table TABLE#DATA#SITRN_DBF and can be linked either to the trade or to the payments.

4.2.2.1 – In Mx G2000

For the trades, there is a link from TRN_HDR_DBF to the table SI_GRID_DBF then to the SI table.

For the payments, fields M_VOSTRO_SI and M_NOSTRO_SI point directly to the SI table.

4.2.2.2 – In Mx.3

For the trades, it is now necessary to go threw the table TRN_EXT_DBF as trades are versioned thanks to the join TRN_EXT_DBF.M_REFERENCE = SI_GRID_DBF.M_TRN_NB.

For the deliverables, fields M_ROUTE and M_VOS_ROUTE in the table DLV_CASH_DBF point directly to the SI table.

4.2.3 – Fields mapping

- Table SI_GRID_DBF: New fields have been added.

M_PHYS_PROD	Physical product (commodities)
M_LOCATION	Location (commodities)

M_SI_FCI	SI Flow Custom Information, added for future use thus it won't appear in SI display or anywhere else ...(commodities)

- Table SI_KEY_DBF:
    - ♦ The field M_VAL_STATUS has been transferred to field STP_STATUS in the table STPSI_ENTRY_TABLE.
    - ♦ New fields have been added.

M_FLOW_TYPO	Flow typology
M_LEGAL_ENT	Legal entity
M_NATURE	Flow nature (cash, delivery..)
M_PROC_ENT	Processing entity
M_SET_METHOD	Settlement method
M_TYPOLOGY	Component typology
M_USAGE	Contract usage
M_CLEARER	exists in v2000.2.11 but wasn't merged in v3.1 till now (commodities)
M_PHYS_PROD	Physical Product (commodities)
M_LOCATION	Location (commodities)
M_SI_FCI	SI Flow Custom Information, added for future use thus it won't appear in SI display or anywhere else ...(commodities)
M_SI_TCI	SI Trade Custom Information, used as an additional criteria when clients need it, the field will be set in TRN_HDR by completion rules(commodities)

- Table TRN_HDR_DBF: New fields have been added.

M_SI_TCI	SI Trade Custom Information, used as an additional criteria when clients need it, the field will be set in TRN_HDR by completion rules(commodities)

4.3 – Deliverable and SI dynamic table

In Mx G2000, the payment & SI dynamic table is of type "Payment" whereas in Mx.3, the deliverable & SI dynamic table is of type "Deliverable cash".

Consequently, it is not possible to import the payment dynamic table directly from the Mx G2000 environment into the Mx.3 environment.

Here is the mapping:

- For Deliverable fields

2.10 field	2.10 Description	3.1 field	Comments
PF_ACC_SEC	Accounting section	DF_ACCSEC	Accounting section
PF_AMOUNT	Flow amount	DF_QTY	Deliverable amount
PF_AMT_OD	Flow amount (OD)	Not maintained in Mx.3	
PF_AMT_ODR	Flow amount (OD, rounded)	Not maintained in Mx.3	

PF_AMT_R	Flow amount (rounded)	DC_RNDQTY	Deliverable rounded amount
PF_BIRTH	Operation# (birth)	DC_OIBIRTH	Event id that creates the flow
PF_CAL_DA0	Calculation date	DF_PSDATE	Period start date
PF_CAL_DA1	Calculation date	DF_PEDATE	Period end date
PF_CMP_DAT	Flow computer date	DF_CMPDATE	
PF_CMP_TIM	Flow computer time	Not maintained in Mx.3	
PF_COMMENT	Flow comment	DC_COMMENT	Comment
PF_CREDIT	Credit (C or D)	DF_PR	Values P or R. the mapping is: D–>P and C–> R
PF_CTRP	Counterpart	DF_BPARTY	Trade party
PF_CUR	Flow currency	DC_CUR	Deliverable currency
PF_CUR_OD	Flow currency (OD)	Not maintained in Mx.3	
PF_DEATH	Operation# (death)	DC_OIDEATH	Event id that cancels the flow
PF_ENTITY	EntityEntity	DF_ENT or DF_LENT	Closing entity or Legal entity At flow level closing entity = legal entity except for internal deals which don't produce any payments
PF_EVT_ID	Trade event#	Not maintained in Mx.3	
PF_FAMILY	Trn. family	DF_TFAMILY	Transaction family
PF_FIX_DAT	Fixing date	DC_FIXDATE	Fixing date
PF_FTYPE	Flow type	DF_TYPE	Deliverable type
PF_GROUP	Trn. group	DF_TGROUP	Transaction group
PF_ID	Flow#	DF_REF	Deliverable reference
PF_LEG	Leg	DF_LEG	Leg
PF_LTI_ID	Linked trade#	Not maintained in Mx.3	
PF_MANUAL	Manual entry	DF_MAN	Manual
PF_MOP_ID	Market operation#	DF_EVTREF	Event id
PF_NET_AG	Flow netting agreement	DF_NETAG	Deliverable netting agreement
PF_NET_AGT	Trn. netting agreement	DF_TNETAG	Transaction netting agteement
PF_NET_SCA	Scanned by netting	DF_ISSNET	Issued from netting
PF_NOS_DAT	Nostro date	Not maintained in Mx.3	
PF_PHASE	Phase	DF_PHASE	Phase
PF_REL_DAT	Release date	DF_RDATE	Release date
PF_REL_TIM	Release time	Not maintained in Mx.3	
PF_SI_N	Settlement nostro ref	DF_NSIREF	Settlement nostro ref
PF_SI_V	Settlement vostro ref	DF_VSIREF	Settlement vostro ref
PF_SPE_N	Settlement nostro specific (deal or flow)	if DF_RSMN = 1 or 3 then Y else N	Settlement nostro specific
PF_SPE_V	Settlement vostro specific (deal or flow)	if DF_RSMV = 1 or 3 then y elsen'	Settlement vostro specific
PF_STATUS	Flow status	DF_STATUS	Deliverable status
PF_SYS_DAT	Flow system date	DF_DATE	System date
PF_TRN_LNK	Trn# (link)	DC_TRNLNK	Trade number or 0 if the flow is cancelled or reversed.
PF_TRN_NB	Trn# (current)	DC_TRDNB	Trade number
PF_TRN_ONB	Trn# (origin)	DC_TRNID	Trade origin number

PF_TYPE	Trn. type	DF_TTYPE	
PF_TYPO	Trn. typology	DF_TTYPO	
PF_USER	User name	DC_USER	User
PF_VAL_DAT	Flow value date	DF_VDATE	Value date

- For SI fields:

All fields are maintained.

4.4 – Audit tables

4.4.1 – Deliverable audit

In Mx G2000, payments are audited in the table PAY_AUD_DBF.

In Mx.3, there is no equivalent table in Mx.3. It is possible to use the table DLV_EXT_DBF that stored all payment versions. There is a table DLV_AUDIT_DBF but it's not considered as a complete audit table. When a contract is inserted, a line is inserted in DLV_AUDIT_DBF for each deliverable type it produces, according to deliverable settings. When the generation mode is asynchronous this line will be used later (and updated) by the task that generates the deliverables.

4.4.2 – Settlement instructions audit

In Mx G2000 and in Mx.3, SI are audited in the table AUD_SI_DBF.

5 – Processing

5.1 – Confirmation instructions

5.1.1 – Description

Confirmation instructions (CI) are information automatically inherited by the deal at insertion time. They allow an easily configurable and efficient routing of the deal confirmation documents, based on user–definable criteria. The Confirmation instructions module is based on the principle of general case and exception: a set of rules is defined for general cases (assumed to represent the majority of cases), and additional rules are eventually defined to apply to exceptions (for example, for one given counterpart). This allows a quicker and more maintainable configuration of the module.

In Mx.3, there is no confirmation instruction linked to a deliverable.

Therefore,

– a confirmation instruction with a value of 1 for the field M_DATA_TYPE of the CTP_CI_DBF table will not be used by Mx.

– a document type with a value of 1 for the field M_TYPE_PAY of the CTP_DOCT_DBF will not be used by Mx.

5.1.2 – Data model

The confirmation instructions are stored in the CTP_CI_DBF table. As the CI are defined for a given counterparty, a link between the counterparty table and the CI can be used to access to the confirmation instructions.

5.1.2.1 – Datamodel schema in Mx.3

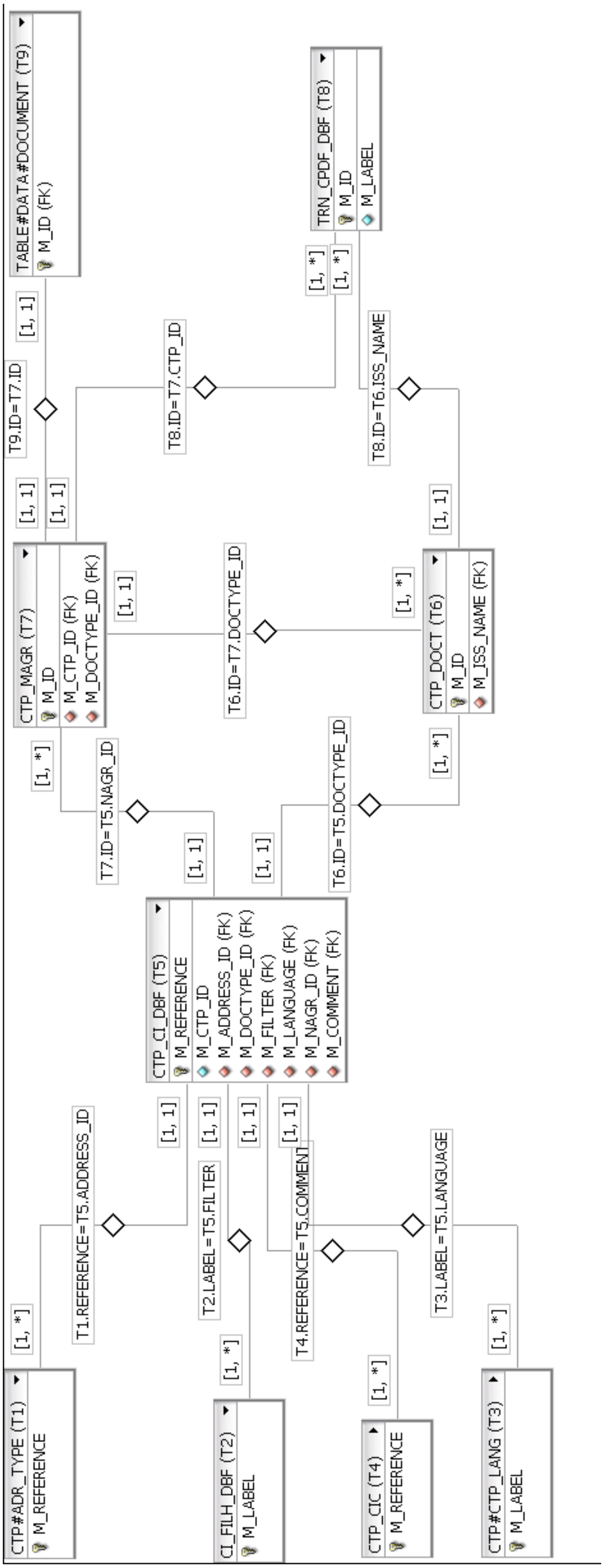

Figure 17: Confirmation instructions data model (v3.1)

5.1.2.2 – Fields Mapping

- Counterparty confirmation instructions: Table CTP_CI_DBF

2.10 Table	2.10 Field	Description	3.1 Table	3.1 Fields
CTP_CI	M_ADDRESS_ID	Type of Address	CTP_CI	M_ADDRESS_ID
CTP_CI	M_COMMENT	Custom Information	CTP_CI	M_COMMENT
CTP_CI	M_CTP_ID	ID of the counterpart	CTP_CI	M_CTP_ID
CTP_CI	M_CTP_TYPE	Type of the counterpart0 if Common1 if Ourselves2 if Specific	CTP_CI	M_CTP_TYPE
CTP_CI	M_DATA_TYPE	Type of the confirmation0 if transaction1 if payment2 if fixing	CTP_CI	M_DATA_TYPE Type of the confirmation0 if transaction1 if payment2 if event as fixing is an event in MX.3
CTP_CI	M_DOCTYPE_ID	Document Type	CTP_CI	M_DOCTYPE_ID
CTP_CI	M_END_DATE	End date of the CI	CTP_CI	M_END_DATE
CTP_CI	M_FILTER	Filter of the CI	CTP_CI	M_FILTER
CTP_CI	M_ID	ID of the CI	CTP_CI	M_ID
CTP_CI	M_LANGUAGE	Language of the CI	CTP_CI	M_LANGUAGE
CTP_CI	M_MAGR_ID	Master Agreement ID	CTP_CI	M_MAGR_ID
CTP_CI	M_START_DATE	Start date of the CI	CTP_CI	M_START_DATE

- Counterparty address type: Table CTP#ADR_TYPE_DBF

2.10 Table	2.10 Field	Description	3.1 Table	3.1 Fields
CTP#ADR_TYPE	M_LABEL	Label of the type of address	CTP#ADR_TYPE	M_LABEL
CTP#ADR_TYPE	M_REFERENCE	Reference of the type of address	CTP#ADR_TYPE	M_REFERENCE

- Counterparty custom informations: Table CTP_CIC_DBF

2.10 Table	2.10 Field	Description	3.1 Table	3.1 Fields
CTP_CIC	M_LABEL	Label of the Custom Information	CTP_CIC	M_LABEL
CTP_CIC	M_REFERENCE	Reference of the Custom Information	CTP_CIC	M_REFERENCE

- Counterparty CI languages: Table CTP#CTP_LANG_DBF

2.10 Table	2.10 Field	Description	3.1 Table	3.1 Fields
CTP#CTP_LANG	M_LABEL	Label of the language of the CI	CTP#CTP_LANG	M_LABEL

- Counterparty Document type: Table CTP_DOCT_DBF

2.10 Table	2.10 Field	Description	3.1 Table	3.1 Fields

CTP_DOCT	M_DESC	Description of the document type	CTP_DOCT	M_DESC
CTP_DOCT	M_END_DATE	End date of the document type	CTP_DOCT	M_END_DATE
CTP_DOCT	M_ID	ID of the document type	CTP_DOCT	M_ID
CTP_DOCT	M_INS_DATE	Insertion date (Server date)	CTP_DOCT	M_INS_DATE
CTP_DOCT	M_INS_TIME	Insertion time (Server time)	CTP_DOCT	M_INS_TIME
CTP_DOCT	M_INS_USER	Insertion User (Login)	CTP_DOCT	M_INS_USER
CTP_DOCT	M_ISS_NAME	Name of the counterpart	TRN_CPDF	M_LABEL: to get the label join with TRN_CPDF using CTP_DOCT.M_ISS_NAME = TRN_CPDF.M_ID
CTP_DOCT	M_ISS_TYPE	0 if Organization1 if Bank2 if Other	CTP_DOCT	M_ISS_TYPE
CTP_DOCT	M_MOD_DATE	Last modification date (Server date)	CTP_DOCT	M_MOD_DATE
CTP_DOCT	M_MOD_TIME	Last modification time (Server time)	CTP_DOCT	M_MOD_TIME
CTP_DOCT	M_MOD_USER	Last modification user (Login)	CTP_DOCT	M_MOD_USER
CTP_DOCT	M_NAME	Name of the document type	CTP_DOCT	M_NAME
CTP_DOCT	M_START_DATE	Start date of the document type	CTP_DOCT	M_START_DATE
CTP_DOCT	M_STATUS	deprecated	deprecated	deprecated
CTP_DOCT	M_TYPE_FIX	1 if fixing 0 else	CTP_DOCT	M_TYPE_FIX
CTP_DOCT	M_TYPE_PAY	1 if payment 0 else	CTP_DOCT	M_TYPE_PAY
CTP_DOCT	M_TYPE_TRN	1 if transaction 0 else	CTP_DOCT	M_TYPE_TRN

- Counterparty Master Agreement: Table CTP_MAGR_DBF

2.10 Table	2.10 Field	Description	3.1 Table	3.1 Fields
CTP_MAGR	M_CTP_ID	ID of the counterpart	CTP_MAGR	M_CTP_ID
CTP_MAGR	M_DESC	Description of the master agreement	CTP_MAGR	M_DESC
CTP_MAGR	M_DOCTYPE_ID	Document type ID	CTP_MAGR	M_DOCTYPE_ID
CTP_MAGR	M_END_DATE	End date of the master agreement	CTP_MAGR	M_END_DATE
CTP_MAGR	M_EXT_SIG	External signature	CTP_MAGR	M_EXT_SIG
CTP_MAGR	M_EXT_SIG_DT	External signature date	CTP_MAGR	M_EXT_SIG_DT
CTP_MAGR	M_ID	ID of the master agreement	CTP_MAGR	M_ID
CTP_MAGR	M_INS_DATE	Insertion date (Server date)	CTP_MAGR	M_INS_DATE
CTP_MAGR	M_INS_TIME	Insertion time (Server time)	CTP_MAGR	M_INS_TIME
CTP_MAGR	M_INS_USER	Insertion User (Login)	CTP_MAGR	M_INS_USER
CTP_MAGR	M_INT_SIG	Internal signature	CTP_MAGR	M_INT_SIG
CTP_MAGR	M_INT_SIG_DT	Internal signature date	CTP_MAGR	M_INT_SIG_DT
None	None	Type of the master agreement	CTP_MAGR	M_MA_TYPE
CTP_MAGR	M_MOD_DATE	Last modification date (Server date)	CTP_MAGR	M_MOD_DATE
CTP_MAGR	M_MOD_TIME	Last modification time (Server	CTP_MAGR	M_MOD_TIME

		time)		
CTP_MAGR	M_MOD_USER	Last modification user (Login)	CTP_MAGR	M_MOD_USER
CTP_MAGR	M_NAME	Name of the master agreement	CTP_MAGR	M_NAME
CTP_MAGR	M_PATH	Path of the master agreement	CTP_MAGR	M_PATH
CTP_MAGR	M_START_DATE	Start date of the master agreement	CTP_MAGR	M_START_DATE
CTP_MAGR	M_STATUS	deprecated	deprecated	deprecated

- User defined Document table: Table TABLE#DATA#DOCUMENT_DBF

2.10 Table	2.10 Field	Description	3.1 Table	3.1 Fields
TABLE#DATA#DOCUMENT	M_ID	ID of the document type	TABLE#DATA#DOCUMENT	M_ID

6 – Trade Financial data

6.1 – Interest rate

6.1.1 – Description

Between versions 2.10 and 3.1, the data model did not change much for interest rates products. Mainly, the changes are linked to enhancement and new developments, such as the addition of the inflation module or some new fields added to grant more flexibility to the configuration of indexes for instance.

The commodity, credit and repo modules share some tables and some fields with the interest rates, therefore, some fields were added that are not filled at all by the rate module. These fields will not be detailed in this document. The purpose of this chapter is to detail the rates new functionalities and the fields they are stored in.

A large number of rates specific fields have also been enhanced to be more precise (stored on more bytes), these changes are not mentioned here either, as most numeric fields have seen their format increased.

6.1.2 – Data model in Mx.3

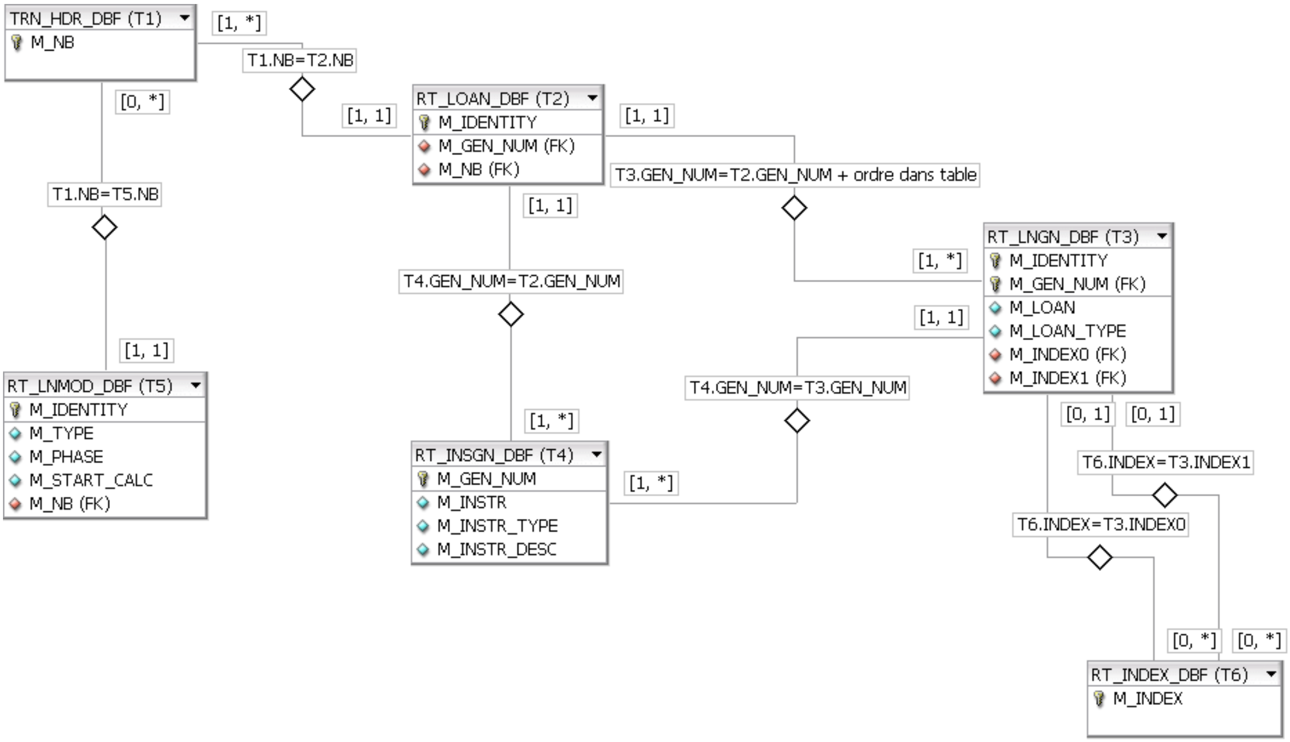

Figure 18: Interest rate deals (v3.1)

6.1.3 – Fields Mapping

- Deal details: Table RT_LOAN_DBF

This table stores details for rate deals, it contains one line per deal having 1 phase and 2 legs. Should a deal have 3 legs and 1 phase, its information would be stored on two lines in this table. This table combined with TRN_HDR and RT_LNGN provides almost all the useful information of a deal.

2.10 Table	2.10 Field	3.1 Table	3.1 Fields	Description
none	none	RT_LOAN	M_AMT_ROLL	When defining an amortizing based on a generator, a roll date can be set, it is stored in this field.
none	none	RT_LOAN	M_CUM_STR	Strike of the target redemption node (strike of the cumulated coupon to reach to kill the deal/leg)
RT_LOAN	M_BAR_TYPE	RT_LOAN	M_BAR_TYPE	Indicates the barrier type : 1 for up and in (>=), 2 for up and out (>=), 3 for down and in (<=), 4 for down and out (<=), 5 for up and in (>), 6 for up and out (>), 7 for down and in (<), and 8 for down and out (<)
None	None	RT_LOAN	M_CORP_MARG	Corporate margin
None	None	RT_LOAN	M_EXR_NAT	Exercise nature, include accruals or no accruals for a bermuda cancellable deal
None	None	RT_LOAN	M_LCALC_DAT	Contains the last calculation date
None	None	RT_LOAN		Contains the first calculation date

			M_SCALC_DAT	
RT_LOAN	M_LG_*	RT_LOAN	M_COMM_DEF	Stores what is common between phases for a multiphase deal. The formula to switch from 2.10 to 3.1 is 1*rate+2*strike+4*paysgn+8*pay.
None	None	RT_LOAN	M_LN_GPAY	Stores the TARN flag (ie bounded condition in Mx)
None	None	RT_LOAN	M_LN_PAY_F2	Stores the secondary payout form of the leg (used in case the deal is set as a collar paying vanilla for the floor part and digital for the strike part for example, this field would store the digital part for the example)

- Indexes: Table RT_INDEX_DBF (new fields)

2.10 Table	2.10 Field	3.1 Table	3.1 Fields	Description
None	None	RT_INDEX	M_CST_FLAG	Linked to the "use constant" flag in the definition off a basket index
None	None	RT_INDEX	M_ECPE, M_ECPE_FREQ, M_ECPE_TYPE, M_ECPE_UNDR	The end date is now stored as a separate date, the fields M_ECPE* store the settings of this field. Those fields work the same as the M_EP*, M_ECP* or M_EI* fields
None	None	RT_INDEX	M_EXCLUDE	For average indexes, somes dates can be excluded now, this fields stores 1 or 0 depending on whether or not some dates are excluded
None	None	RT_INDEX	M_EXCL_GEN	Stores the generator used for the exclusion of dates
None	None	RT_INDEX	M_EXCL_STYLE	Stores whether the exclusion of dates is "inherited from underlying" or "specific"
None	None	RT_INDEX	M_REFERENCE	Reference number of the index
None	None	RT_INDEX	M_IND_LABEL	Index label
None	None	RT_INDEX	M_LOG_DEL	Logical delete date
None	None	RT_INDEX	M_FOLW_FXNG	Pricing functionality. If set to yes, the index is estimated on fixing date + schedule instead of start date + schedule. This field stores this data.
None	None	RT_INDEX	M_LOK_PER	Stores whether or not there is a lockout period defined for the index (only relevant for average or compound indexes)
None	None	RT_INDEX	M_LOK_PER_SH	Lockout period shifter
None	None	RT_INDEX	M_INTRADAY	Only for generic indexes
None	None	RT_INDEX	M_RE_CL_M	The calendar followed by the calculation start and end dates can now be defined. There is a setting in the general settings that can be overwritten at the index definition level. This fields contains the "inherited"/ "redefined" information
None	None	RT_INDEX	M_RE_ID_CL	If the previous field is set to redefined, this field stores the "index calendar" information (calculation end date)
None	None	RT_INDEX	M_RE_ST_CL	If the previous field is set to redefined, this field stores the "start calendar" information (calculation start date)
None	None	RT_INDEX	M_INFLTYPE	Relevant for inflation indexes only, stores the inflation type, growth rate or inflation curve

None	None	RT_INDEX	M_FREQUENCY	Stores the fixing frequency of the inflation index (usually monthly fixing)
None	None	RT_INDEX	M_FIXANN	Fixing anniversary, date of the monthly fixing (usually the first day of the month)
None	None	RT_INDEX	M_INTPL_BN	For inflation interpolation, this field stores the numerator to use
None	None	RT_INDEX	M_INTPL_BD	For inflation interpolation, this field stores the denominator to use

- Rate deal generation: Table RT_LNGN_DBF (new fields)

2.10 Table	2.10 Field	3.1 Table	3.1 Fields	Description
None	None	RT_LNGN	M_BROK_COND	The type of broken period can now be set to conditional, this field stores the name of the defined condition.
None	None	RT_LNGN	M_FOLW_FXNG	Pricing functionality. If set to yes, the index is estimated on fixing date + schedule instead of start date + schedule. This field stores this data.
None	None	RT_LNGN	M_LEV_MODE	Leg evaluation mode
None	None	RT_LNGN	M_MARG_MODE	Margin mode (0=additive, 1=multiplicative, 2=In underlying)
None	None	RT_LNGN	M_SIMP_SCH	Stores if the schedule is a single period one.
None	None	RT_LNGN	M_DLVTP	Indexation type of the initial capital (currency, commodity, equity, bond)
None	None	RT_LNGN	M_FINALTP	Indexation type of the final capital (currency, commodity, equity, bond)
None	None	RT_LNGN	M_INTRMTP	Indexation type of the intermediate capital (currency, commodity, equity, bond)
None	None	RT_LNGN	M_INTRSTP	Indexation type of the interest flows (currency, commodity, equity, bond)
None	None	RT_LNGN	M_EAC	Capital calculation schedule or shifter
None	None	RT_LNGN	M_EAC_FREQ	Capital calculation schedule frequency
None	None	RT_LNGN	M_EAC_TYPE	Capital calculation schedule type
None	None	RT_LNGN	M_EAC_UNDRL	Schedule on which the capital calculation schedule is based
None	None	RT_LNGN	M_EAP	Capital payment schedule or shifter
None	None	RT_LNGN	M_EAP_FREQ	Capital payment schedule frequency
None	None	RT_LNGN	M_EAP_TYPE	Capital payment schedule type
None	None	RT_LNGN	M_EAP_UNDRL	Schedule on which the capital payment schedule is based
None	None	RT_LNGN	M_ECPE	Calculation end schedule or shifter
None	None	RT_LNGN	M_ECPE_FREQ	Calculation end schedule frequency
None	None	RT_LNGN	M_ECPE_TYPE	Calculation end schedule type
None	None	RT_LNGN	M_ECPE_UNDRL	Schedule on which the calculation end schedule is based
None	None	RT_LNGN	M_RC_APPF	Rate conversion applied to rate factor
None	None	RT_LNGN	M_RTF_MODE	Stores whether the rate factor is applied before or after

				the conversion rule
None	None	RT_LNGN	M_PROTZ	Day count fraction (should be set to no for inflation generators only)
None	None	RT_LNGN	M_RSHIFT	Fixing lag of the return box (for inflation generators)
None	None	RT_LNGN	M_RETDSCEV	Return evaluation mode
None	None	RT_LNGN	M_RETINRP	Inflation setting : return interpolation mode (linear, loglinear or piecewise)
None	None	RT_LNGN	M_RETINT	Return type (spread, ration, return; based on origin or previous reference)

- Rate details generator : Table RT_INSGN_DBF

2.10 Table	2.10 Field	3.1 Table	3.1 Fields	Description
None	None	RT_INSGN	M_ACC_MOD	Stores the accrual delay mode (+1OD or redefined), mainly used for call deposits. Used for swap which evaluation has been set to accruals
None	None	RT_INSGN	M_ACC	If the previous flag was set to redefined, this field stores the accrual delay
None	None	RT_INSGN	M_FLP_MOD	Only for call deposits, flow projection mode (+1OD or redefined)
None	None	RT_INSGN	M_FLP	If the previous flag was set to redefined, this field stores the flow projection delay
None	None	RT_INSGN	M_BUY_SIGN	For 2 legs deals, stores 0 for "pay fixed" and 1 for "receive fixed"
None	None	RT_INSGN	M_LOG_DEL	Date when the generator was logically deleted
None	None	RT_INSGN	M_SETTL_MOD	Stores if the settlement delay follow he currency or is redefined
None	None	RT_INSGN	M_SETTL	If the previous flag was set to redefined, this field stores the settlement delay

- Deal customizations: Table RT_LNMOD_DBF

2.10 Table	2.10 Field	3.1 Table	3.1 Fields	Description
None	None	RT_LNMOD	M_CHAR	Stores the customization if it is a character chain customization
RT_LNMOD_DBF	M_TYPE	RT_LNMOD	M_TYPE	The code of the various customizations has changed, you can find the new 3.1 code in the appendixes "customization codification"

7 – Miscellaneous

7.1 – About parser functions

- STR()

The parser function STR round down in Mx G2000.

The parser function STR round up in Mx.3.

- X_TRNSI()

The parameter of the function should be the component extension in Mx.3 instead of the trade number in Mx G2000.

In Mx G2000, the parser function X_TRNSI is used in the following format:

X_TRNSI (NB,F_CURRENCY+':D:N','My field')

where

NB – Refers to the transaction number

F_CURRENCY – Refers to the flow currency

D:N – Refers to Debit/Credit and Nostro/Vostro

My field – Refers to one of the UDFs configured in the SI table on which the query is made

In Mx.3, the parser function X_TRNSI will have to use the reference number M_REFRENCE in the TRN_EXT_DBF instead of M_NB in TRN_HDR_DBF.

If used in a dynamic table (the field CMP_EXT, component extension should be checked and the parser function will take the following form:

X_TRNSI (CMX_EXT,F_CURRENCY+':D:N','My field')

If used anywhere else, the parser function will take the following form:

X_TRNSI (TRN_EXT.REFERENCE,F_CURRENCY+':D:N','My field')

- DT_SKIP() and DT_XSKIP()

In Mx G2000, if these functions take as a parameter an inexistent calendar, the date is shifted without taking into account week−ends or holidays.

In Mx.3, using a missing calendar is no more tolerated and the system throws an exception with the calendar label.

When migrating reports, it's possible to reproduce the original behavior in two ways: replace in the parser function the calendar name with "" or add in the database an empty calendar (no week−end, no holidays) having the same label. Note that it's also possible to correct the formula by replacing the missing calendar with a valid one, but this solution might alter the output.

- Horizontal formula

When building a horizontal field at the dynamic table level, empty strings are now identified using the expression =''.

7.2 – About other data model tables

- Flex tables: RTBLOCK#RTBK*

All flex tables have not changed.

- The table: RTRN_HDR_DBFT

This table was a subset of TRN_HDR_DBF in Mx G2000. This table is no more used in Mx.3.

7.3 – About Mreport

Table relations

In a table relation, if you have a 1−n relation in the data model and if in Mreport a 1−1 join is confugred instead, Mreport will return onlt the first corresponding line physically stored in the table. After migration, order of lines can be modified so the line returned by the report can be different. The report should be redsigned as it is consider as a wrong conception of the report.

Also, 1−n relation can not be configured in Mreport in Mx G2000 as well as in Mx.3.

7.4 – Dynamic table fields

7.4.1 – Field Format

Some fields format has been modified between Mx G2000 and Mx.3. As the size of the field should be the same for external systems, note that they could be truncated as they will be resized at the level of the report. Here is a list of the concerned fields:

Table and field	Format in V2.10	Format in V3.1
TP_CNTRP	15	35

7.4.2 – Dynamic table removed fields

Dynamic table field	Comment
F_FIX	Never filled in 2.10, can be replaced with horizontal field of type C and value " "
F_FIXPRIC	Never filled in 2.10, can be replaced with horizontal field of type N and value 0

7.4.3 – New behavior of dynamic table fields

- TRNRP_DT

-
    - ♦ DT_CAPCUR0, DT_CAPREM0, DT_START0 and DT_END0: are filled for capital exchange (when DT_FLOWNAT='PRI') in Mx.3 whereas they were empty in Mx G2000.
    - ♦ DT_FLOWTYP: This field contains the cash flow type. The value of this field is filled for all products since Mx G2000 version, 2.11.32. DT_FLOWTYP is filled by "INT" for interest flows (when DT_FLOWNAT=INT) in Mx.3 whereas it was empty in Mx G2000.
    - ♦ DT_FLOWNAT: In 2.10, instead of 'CAS' typology, flows coming from a Restructure have PRI or FLW in the DT table. In Mx.3 DT_FLOWNAT returns now th correct value 'CAS'.

- TRNRP_CS

to be completed

- Others

In 2.10, Amortizing flows are usually linked to a default leg (=0), event though the right leg is 1. In Mx.3, flows are correctly linked to the leg 1.

- Trades fields
  - TP_CP: In dynamic tables, the field TP_CP is now filled for EQD/BOND/IDX.
  - TP_BROKER: contains an id, to join with the counterpart table.

7.5 – Description of TRN_HDR_DBF fields

Here is a description of TRN_HDR_DBF fields in Mx.3

Fields	Description	Status
M_NB	Trade number	
M_LTI_NB	Link trade number	Not maintained in Mx.3
M_GID	Global id	
M_NB_TISTAMP	Timestamp number	
M_TRN_FMLY	Trade family	
M_TRN_GRP	Trade Group	
M_TRN_TYPE	Trade type	
M_TRN_GTYPE	Trade type id	
M_TRN_TYPO	Trade typology	Not maintained in Mx.3; The replacement of this field should be discussed on a case by case basis for each customer.
M_INSTRUMENT	Instrument	
M_RSKSECTION	Risk section	
M_PL_INSCUR	PL instrument currency	
M_PL_KEY1	PL key	
M_MKT_LABEL	Market label	
M_MKT_INDEX	Market index	
M_CNS_ACTIVE	Used for consolidation	
M_AGREED_TRN	Trade agreement	Not maintained in Mx.3
M_REAL	if draft deal, M_REAL=1	Not maintained in Mx.3
M_COMMENT_BS	buy/sell	
M_CLIENT	Client	Not maintained in Mx.3
M_BINTERNAL	Buy internal Y/N	
M_BTRADER	Buy trader	
M_BPFOLIO	Buy portfolio (M_LABEL)	
M_BCOMMENT0	Buy comment 0	
M_BCOMMENT1	Buy comment 1	
M_BCOMMENT2	Buy comment 2	
M_BSTRATEGY	Buy strategy	

M_BSECTION	Buy accounting section	
M_BENTITY	Buy closing entity	
M_SINTERNAL	Sell internal	
M_STRADER	Sell trader	
M_SPFOLIO	Sell portfolio (M_LABEL)	
M_SCOMMENT0	Sell comment 0	
M_SCOMMENT1	Sell comment 1	
M_SCOMMENT2	Sell comment 2	
M_SSTRATEGY	Sell strategy	
M_SSECTION	Sell accounting section	
M_SENTITY	Sell closing entity	
M_TRN_STATUS	Trade status (LIVE/DEAD/MOP)	
M_TRN_DATE	Trade date	
M_TRN_TIME	Trade time	
M_TRN_EXP	Trade expiry date	
M_SYS_DATE	system dare at trade insertion	
M_QTY_NB		Not maintained in Mx.3
M_CREATOR	Creator number	
M_CRE_CMMOUT		Not maintained in Mx.3
M_NB_EXT	Trade external number	
M_BO_SGN	Bo signature	Not maintained in Mx.3
M_BO_CMT	Bo comitor	Not maintained in Mx.3
M_BO_CNF	Bo confirmation	Not maintained in Mx.3
M_ACC_PROR	Accounting field	
M_RPL_AMO	Accounting field	
M_RPL_AMTTYP	Accounting field	
M_RPL_AMT	Accounting field	
M_RPL_CUR	Accounting field	
M_RPL_USRDAT	Accounting field	
M_RPL_DATE1	Accounting field	
M_RPL_DATE2	Accounting field	
M_UPL_FLAG	Accounting field	
M_UPL_MODE	Accounting field	
M_UPL_AMTEVC	Accounting field	
M_UPL_AMT	Accounting field	
M_UPL_AMTDIS	Accounting field	
M_IRV_TYPE	Accounting field	
M_IRV_AMT	Accounting field	
M_FCP_REVAL	Accounting field	
M_HEDGE_FLAG	Hedge flag	
M_HEDGED_ID	Hedge id	
M_HEDGED_MAT	Hegde maturity	
M_PAY_NET	Payment Netting agreement	
M_VAL_STATUS	old validation status, replaced by STP_STATUS	Not maintained in Mx.3
M_OLK		Not maintained in Mx.3

M_MAIN		Not maintained in Mx.3
M_FLOW_FLAG	User flow flag	
M_AREA_CODE	Area code	
M_MRPL_DATE	Actual date (always filled)	
M_MRPL_ONB	Original trade number (always filled)	
M_CAN_GXIT	Candidate to global temination/expiry	
M_GXIT_DATE	date of the global expiry	
M_NB_AMD		Not maintained in Mx.3
M_DTE_AMD		Not maintained in Mx.3
M_MOP_LAST		Not maintained in Mx.3
M_MOP_CREAT		Not maintained in Mx.3
M_MOP_CRSUB		Not maintained in Mx.3
M_CNS_LOADFO		Not maintained in Mx.3
M_BRK_THIRDP	Third party brokerage	
M_OPT_FLWFST	Date of the first trade flow	
M_OPT_FLWLST	Date of the last trade flow	
M_OPT_ACCLST	Accounting field	
M_OPT_MOPFST		Not maintained in Mx.3
M_OPT_MOPLST		Not maintained in Mx.3
M_OPT_MOPLSD		Not maintained in Mx.3
M_OPT_MOPCNT		Not maintained in Mx.3
M_OPT_MOPSUB		Not maintained in Mx.3
M_OPT_MOPNB		Not maintained in Mx.3
M_BRW_NOM1	Nominal	
M_BRW_NOMU1	Underlying (e.g. currency)	
M_BRW_NOM2	Other nominal	
M_BRW_NOMU2	Other underlying	
M_BRW_RTE1	Rate leg1	
M_BRW_RTE2	Rate leg2	
M_BRW_MRG1	Margin leg 1	
M_BRW_MRG2	Margin leg2	
M_BRW_STRK	Strike	
M_BRW_CP	Call/Put	
M_BRW_AE	American/Europoan	
M_BRW_PR1	Pay/Receive leg1	
M_BRW_PR2	Pay/Receive leg2	
M_BRW_FV1	Fix/Floating leg1	
M_BRW_FV2	Fix/Floating leg2	
M_BRW_SDTE	Start date	
M_BRW_ODPL	CMM field	Not maintained in Mx.3
M_BRW_ODNC0	CMM fieldNot maintained in Mx.3	Not maintained in Mx.3
M_BRW_ODNC1	CMM field	Not maintained in Mx.3
M_BRW_ODFC0	CMM field	Not maintained in Mx.3
M_BRW_ODFC1	CMM field	Not maintained in Mx.3
M_OPT_CMMNAT		Not maintained in Mx.3

M_OPT_CMMDTE		Not maintained in Mx.3
M_OPT_CMMCUR		Not maintained in Mx.3
M_OPT_CMMSTL		Not maintained in Mx.3
M_OPT_CMMSPD		Not maintained in Mx.3
M_OPT_CMMSPS		Not maintained in Mx.3
M_PURGE_DATE		Not maintained in Mx.3
M_SALES	Sales (user)	
M_STS_FLAG		Not maintained in Mx.3
M_LAT	Acceptance flag	
M_HOST_SYS	Hosting system	
M_CNT_INTID	Contract class id	
M_CONTRACT	Contract number	
M_OPT_NATURE	Nature of th option	
M_ORIG_PS_ID	Origin publishing system id	
M_PS_ID	Publishing system id	
M_STAT_CAT	Statement category (Accounting)	
M_CUSTOM	Binary code to identify custmized/non custo fields	
M_COMP_TYPO	Code to identify trade as part of contract	
M_TYPOLOGY	Trade typology (what's this?), used in right matrice	
M_USAGE	Usage (What's for?), used in right matrix	
M_SETTL_METH	Settlement method	
M_P_ENTITY	Processing center entity	
M__DT_TS	trade insertion time stamp	
M_OPT_STSVER	Last trade insertion	
M_PURGE_REF	Logical purge id	
M_SRC_PFOLIO	Source portfolio (M_ID)	
M_DST_PFOLIO	Destination portfolio (M_ID)	
M_COUNTRPART	Destination counterparty	
M_PURPOSE	Trade purpose (following class id MxContractTradePurpose)	
M_BLENTITY	Buy legal entity	
M_SLENTITY	Sell legal entity	

7.6 – UDF table impact overview

UDF Tables	Impact inV3.1
TABLE#DATA#DEALCOM_DBF.M_NB	Impacted by versioning
TABLE#DATA#DEALCRD_DBF.M_NB	Impacted by versioning
TABLE#DATA#DEALCURR_DBF.M_NB	Impacted by versioning
TABLE#DATA#DEALEQD_DBF.M_NB	Impacted by versioning
TABLE#DATA#DEALFIN_DBF.M_NB	Impacted by versioning
TABLE#DATA#DEALFXD_DBF.M_NB	Impacted by versioning
TABLE#DATA#DEALHYB_DBF.M_NB	Impacted by versioning
TABLE#DATA#DEALIRD_DBF.M_NB	Impacted by versioning

TABLE#DATA#DEALSCF_DBF.M_NB	Impacted by versioning
TABLE#DATA#PAYFLOW_DBF.M_FLOW_ID	Deprecated/Removed
TABLE#DATA#LINKEDTR_DBF.M_REF	Deprecated/Removed
TABLE#DATA#MARKETOP_DBF.M_NB	Deprecated/Removed
TABLE#DATA#PORTFOLI_DBF.M_LABEL	Valid
TABLE#DATA#SECISSUE_DBF.M_SNAME	Valid but not confirmed yet
TABLE#DATA#SECURITI_DBF.M_SE_LABEL	Valid but not confirmed yet
TABLE#DATA#SITRN_DBF.M_REF	Impacted by versioning
TABLE#DATA#STRATEGY_DBF.M_LABEL	Deprecated/Removed
TABLE#DATA#TRADEEVE_DBF.M__KEY_	Deprecated/Removed
TABLE#DATA#PCK_****	Deprecated/Removed
TABLE#DATA#CREDITFI_DBF.M_NB	Valid but not confirmed yet

Appendix 1 – Compatibility views

To activate the compatibility module that allows the use of the compatibility tables (TRN_DD_DBF and DLV_CDD_DBF) and the use of the compatibility views, it is mandatory to add the default command /DD_COMPATIBILITY in the launcherall.mxres file.

If the compatibility flag is set, compatibility views are created and managed by the code during the install/install procedure.

The SQL queries are generated automatically for the four views built on UDF tables. This way, views can be updated accordingly when modifications on user definable fields occurred.

For other views, the install/install procedure will creates murex compatibility views declared in the file fs\murex\mxres\common\dbconfig\dbviews.mxres.

It is not recommended to use views automatically. Although views do facilitate the migration of the report, they introduce performance problems and complexity to the report.

Below, the SQL of each view is detailed.

1.1 – UDF tables views

This section provides samples of SQL queries for generating UDF table views.

Click here (see file example 1 : tables_data_dealIRD_vw_db.txt) to see TABLE#DATA#DEALIRD_VW

Click here (see file example 2 : tables_data_dealcurr_vw_db.txt) to see TABLE#DATA#DEALCURR_VW

Click here (see file example 3 : tables_data_dealEQD_vw_db.txt) to see TABLE#DATA#DEALEQD_VW

Click here (see file example 4 : tables_data_dealCRD_vw_db.txt) to see TABLE#DATA#DEALCRD_VW

1.2 – Market operations views

This section provides samples of SQL queries for generating Market operation table views.

1.2.1 – MKT_OP_VW, without any settlement information

Sample here (see file example 5 : mkt_op_vw_dbf.txt) .

1.2.2 – MKT_OP_ONES_VW, with information for one of settlements

Sample here (see file example 6 : mkt_op_ones_vw_dbf.txt) .

1.2.3 – MKT_OP_ALLS_VW, with information for all settlements

Sample here (see file example 7 : mkt_op_alls_vw_dbf.txt) .

1.3 – Other views

1.3.1 – Additional flows view TRN_HDRF_VW

Sample here (see file example 8 : trn_hdrf_vw_dbf.txt)

1.3.2 – Dates view PROCESS#PS_DATE_VW

Sample here (see file example 9 : process_ps_date_vw_dbf.txt) .

Appendix 2 – Customization codification

Code	customization
0	Payment date
1	Fixing date
2	Calculation start date
3	Capital date
4	Interest date
5	Interest flow
6	Remaining capital payment flow
7	Main strike
8	Fixing
9	Period main volatility
10	Calculation end date
11	Interest margin
12	Secondary strike
13	Period secondary volatility
14	First fixing
15	Exercise date
16	Compounding rule
17	Compounding mode
18	Settlement date
19	Settlement rate
20	Settlement flow
21	Settlement calculation start date
22	Settlement calculation end date
23	Adjustment rule
24	Interest flow factor
25	Capital calculation start date
26	Capital calculation end date
27	Capital payment date
28	Capital payment flow
29	Exercised (for flex)
30	Cliquet strike
31	Cliquet secondary strike
32	Amortizing annuity
33	Barrier
34	Capital payment flow
35	Capital calculation payment flow
36	Interest reinvestment
37	Interest rate + dividend
38	Interest flow + dividend flow
39	Interest first indexation
40	Interest second indexation

41	Fixing first indexation
42	Fixing second indexation
43	Margin first indexation
44	Margin second indexation
45	Interest rate first indexation
46	Interest rate second indexation
47	Initial capital first indexation
48	Initial capital second indexation
49	Capital flow first indexation
50	Capital flow second indexation
51	Final capital first indexation
52	Final capital second indexation
53	Remaining capital first indexation
54	Remaining capital second indexation
55	Interest first indexation date
56	Interest second indexation date
57	Fixing first indexation date
58	Fixing second indexation date
59	Margin first indexation date
60	Margin second indexation date
61	Interest rate first indexation date
62	Interest rate second indexation date
63	Initial capital first indexation date
64	Initial capital second indexation date
65	Capital flow first indexation date
66	Capital flow second indexation date
67	Final capital first indexation date
68	Final capital second indexation date
69	Remaining capital first indexation date
70	Remaining capital second indexation date
71	Rate factor
72	First level underlying index fixing
73	Second level underlying index fixing
74	First level underlying index weight
75	Second level underlying index weight
76	Strike first indexation
77	Strike second indexation
78	Strike first indexation date
79	Strike second indexation date
81	Digital rate
82	Dividends first indexation
83	Dividends second indexation
84	Dividends first indexation date
85	Dividends second indexation date
86	Tax credit first indexation

87	Tax credit second indexation
88	Tax credit first indexation date
89	Tax credit second indexation date
90	Exercise underlying start date
92	Applied capital flow
93	Quantity
94	Revolving forward
95	Revolving backward
96	Revolving accrual payment
97	Revolving accrual shifter
98	Non par repayment
99	Initial capital payment date
100	Final capital payment date
101	Interest reinvestment no capital payment
102	No capital payment
104	Standard amount FX factor
105	End date second indexation date
106	Nth to default (credit)
107	Underlying first date
108	Underlying second date
109	Exercise start shifter
111	Sales margin
112	Applied remaining capital flow
113	Range first strike
114	Range second strike
115	Cumulative strike
116	Flat correlation
118	Ratchet first factor
119	Ratchet second factor
120	Ratchet third factor
121	Ratchet first margin
122	Ratchet second margin
123	Ratchet third margin
124	Exercise strike
125	Base correlation inf
126	Base correlation sup
127	Compound correlation
128	Second barrier
129	Secondary digital rate
130	Secondary strike first indexation
131	Secondary strike second indexation
132	Secondary strike first indexation date
133	Secondary strike second indexation date
135	Exercise penalty fee rate
136	Exercise rental fee rate

137	Exercise fee capital mode
138	Exercise fee end date mode
139	Exercise fee end date specific
140	Exercise fee end date shifter
141	Basket computation type
142	Secondary range first strike
143	Secondary range second strike
145	Fixing compounding
146	No amortizing
147	Amortizing end date
148	Quantity price
149	Revolving event date
150	Schedules limit date
151	Accrual capital flow
152	Ex dividend period first date
153	Ex dividend period last date

File example 1 : tables_data_dealIRD_vw_db.txt

```
Create view TABLE#DATA#DEALIRD_VW_DBF
as
     select EXT.TIMESTAMP, EXT.M_IDENTITY,
     EXT.M_TRADE_REF as M_NB,UDF.M_USR_FIELD1,UDF.M_USR_FIELD2,...
     from
     TRN_EXT_DBF     EXT,
     TABLE#DATA#DEALIRD_DBF UDF,
     TRN_HDR_DBF HDR,
     CONTRACT_DBF CON
     where
     HDR.M_CONTRACT = CON.M_REFERENCE
     and CON.M_VERSION = EXT.M_VERSION
     and EXT.M_TRADE_REF = HDR.M_NB
     and EXT.M_UDF_REF = UDF.M_NB
```

File example 2 : tables_data_dealcurr_vw_db.txt

```
create view TABLE#DATA#DEALCURR_VW_DBF
as
     select EXT.M_TRADE_REF as M_NB,UDF.M_USR_FIELD1,UDF.M_USR_FIELD2,...
     from
     TRN_EXT_DBF     EXT,
     TABLE#DATA#DEALCURR_DBF UDF,
     TRN_HDR_DBF HDR,
     CONTRACT_DBF CON
     where
     HDR.M_CONTRACT = CON.M_REFERENCE
     and CON.M_VERSION = EXT.M_VERSION
     and EXT.M_TRADE_REF = HDR.M_NB
     and EXT.M_UDF_REF = UDF.M_NB
```

File example 3 : tables_data_dealEQD_vw_db.txt

```
create view TABLE#DATA#DEALEQD_VW_DBF
as
     select EXT.TIMESTAMP, EXT.M_IDENTITY,
     EXT.M_TRADE_REF as M_NB, UDF.M_USR_FIELD1,UDF.M_USR_FIELD2,...
     from
     TRN_EXT_DBF     EXT,
     TABLE#DATA#DEALEQD_DBF UDF,
     TRN_HDR_DBF HDR,
     CONTRACT_DBF CON
     where
     HDR.M_CONTRACT = CON.M_REFERENCE
     and CON.M_VERSION = EXT.M_VERSION
     and EXT.M_TRADE_REF = HDR.M_NB
     and EXT.M_UDF_REF = UDF.M_NB
```

File example 4 : tables_data_dealCRD_vw_db.txt

```
create view TABLE#DATA#DEALCRD_VW_DBF
as
     select EXT.TIMESTAMP, EXT.M_IDENTITY,
     EXT.M_TRADE_REF as M_NB,UDF.M_USR_FIELD1,UDF.M_USR_FIELD2,...
     from
     TRN_EXT_DBF     EXT,
     TABLE#DATA#DEALCRD_DBF UDF,
     TRN_HDR_DBF HDR,
     CONTRACT_DBF CON
     where
     HDR.M_CONTRACT = CON.M_REFERENCE
     and CON.M_VERSION = EXT.M_VERSION
     and EXT.M_TRADE_REF = HDR.M_NB
     and EXT.M_UDF_REF = UDF.M_NB
```

File example 5 : mkt_op_vw_dbf.txt

```
  create view MKT_OP_VW_DBF
  as
       select
       EXT.TIMESTAMP,
       EXT.M_IDENTITY,
       EVT.M_REFERENCE as M_NB,
       EXT.M_TRADE_REF,
       EXT.M_REFERENCE as M_EXTREF,
       EXT.M_DATE,
       EVT.M_COMMENT,
       EVT.M_USER as M_TRADER,
       EXT.M__DT_TS as M_SYS_DATE,
       case
        when EXT.M_EVT_INTID = '1.220' then 'RPL_D' --Cancel
        when EXT.M_EVT_INTID = '1.372' then 'RPL_M' --Cancel & Re-issue
        when EXT.M_EVT_INTID = '1.373' or EXT.M_EVT_INTID = 'MizOp51273' then 'RPL' --
>  or Assignment
        when EXT.M_EVT_INTID = '1.589' then 'EXP' --Expiry
        when EXT.M_EVT_INTID = '1.371' then 'XIT' --Unwind (termination)
        when EXT.M_EVT_INTID = 'MwDcj67841' then 'EXR' --Exercise
        when EXT.M_EVT_INTID = 'MJgYK37904' then 'NET' --Netting
        else EXT.M_EVT_INTID
       end as M_TYPE,
       case
        when EXT.M_EVT_INTID = 'MizOp51273' then 'ASSIG' --Assignment
        else ''
       end as M_TYPE_SUB,
       case EXT.M_LAT
        when 0 then 'N'
        else 'Y'
       end as M_LAT,
       IMP.M_HDG_SOLD
       from
       TRN_DD_DBF DD, TRN_EXT_DBF EXT, EVT_EVENT_DBF EVT, EVT_IMP_DBF IMP
       where
       DD.M_LIMPEXTREF? *= EXT.M_REFERENCE
       and EXT.M_EVT_REF *= EVT.M_REFERENCE
       and EXT.M_EVT_VS *= EVT.M_VERSION
       and DD.M_LIMPREF *= IMP.M_REFERENCE
```

File example 6 : mkt_op_ones_vw_dbf.txt

```
  create view MKT_OP_ONES_VW_DBF
  as
       select
       EXT.TIMESTAMP,
       EXT.M_IDENTITY,
       EVT.M_REFERENCE as M_NB,
       EXT.M_TRADE_REF,
       EXT.M_DATE,
       EVT.M_COMMENT,
       EVT.M_USER as M_TRADER,
       EXT.M__DT_TS as M_SYS_DATE,
       case
        when EXT.M_EVT_INTID = '1.220' then 'RPL_D' --Cancel
        when EXT.M_EVT_INTID = '1.372' then 'RPL_M' --Cancel & Re-issue
        when EXT.M_EVT_INTID = '1.373' or EXT.M_EVT_INTID = 'MizOp51273' then 'RPL' -- Restructure or Assi
> gnment
        when EXT.M_EVT_INTID = '1.589' then 'EXP' --Expiry
        when EXT.M_EVT_INTID = '1.371' then 'XIT' --Unwind (termination)
        when EXT.M_EVT_INTID = 'MwDcj67841' then? 'EXR' --Exercise
        when EXT.M_EVT_INTID = 'MJgYK37904' then? 'NET' --Netting
        else EXT.M_EVT_INTID
       end as M_TYPE,
       case
        when EXT.M_EVT_INTID = 'MizOp51273' then 'ASSIG' --Assignment
        else ''
       end as M_TYPE_SUB,
       case EXT.M_LAT
        when 0 then 'N'
        else 'Y'
       end as M_LAT,
       IMP.M_HDG_SOLD,
       FLOW.M_VAL_DATE as M_NFAMOUNTD,
       FLOW.M_AMOUNT as M_NFAMOUNT,
       FLOW.M_CURRENCY,
       case FLOW.M_INTERNAL
         when 0 then CTP.M_DSP_LABEL
         else PTF2.M_DSP_LABEL
       end as M_DEST,
       PTF1.M_DSP_LABEL as M_ORIGIN
       from
       TRN_DD_DBF DD, TRN_EXT_DBF EXT, EVT_EVENT_DBF EVT,
       (       select M_EVT, M_COMPONENT, M_VAL_DATE, M_CURRENCY, M_AMOUNT, M_DEST, M_SOURCE, M_INTERNAL
               from FCE_FLOW_DBF FLOW
               group by M_EVT, M_COMPONENT
               having M_REFERENCE = max(M_REFERENCE)) FLOW,
       TRN_PFLD_DBF PTF2, TRN_CPDF_DBF CTP, TRN_PFLD_DBF PTF1, EVT_IMP_DBF IMP
       where
       DD.M_LIMPEXTREF *= EXT.M_REFERENCE
       and EXT.M_EVT_REF *= EVT.M_REFERENCE
       and EXT.M_EVT_VS *= EVT.M_VERSION
       and EVT.M_REFERENCE *= FLOW.M_EVT
       and EXT.M_TRADE_REF *= FLOW.M_COMPONENT
       and FLOW.M_DEST *= PTF2.M_LABEL
       and FLOW.M_DEST *= CTP.M_LABEL
       and FLOW.M_SOURCE *= PTF1.M_LABEL
       and DD.M_LIMPREF *= IMP.M_REFERENCE
```

File example 7 : mkt_op_alls_vw_dbf.txt

```
  create view MKT_OP_ALLS_VW_DBF
  as
       select
       EXT.TIMESTAMP,
       EXT.M_IDENTITY,
       EVT.M_REFERENCE as M_NB,
       EXT.M_TRADE_REF,
       EXT.M_DATE,
       EVT.M_COMMENT,
       EVT.M_USER as M_TRADER,
       EXT.M__DT_TS as M_SYS_DATE,
       case
        when EXT.M_EVT_INTID = '1.220' then 'RPL_D' --Cancel
        when EXT.M_EVT_INTID = '1.372' then 'RPL_M' --Cancel & Re-issue
        when EXT.M_EVT_INTID = '1.373' or EXT.M_EVT_INTID = 'MizOp51273' then 'RPL' --
>  or Assignment
        when EXT.M_EVT_INTID = '1.589' then 'EXP' --Expiry
        when EXT.M_EVT_INTID = '1.371' then 'XIT' --Unwind (termination)
        when EXT.M_EVT_INTID = 'MwDcj67841' then 'EXR' --Exercise
        when EXT.M_EVT_INTID = 'MJgYK37904' then 'NET' --Netting
        else EXT.M_EVT_INTID
       end as M_TYPE,
       case
        when EXT.M_EVT_INTID = 'MizOp51273' then 'ASSIG' --Assignment
        else ''
       end as M_TYPE_SUB,
       case EXT.M_LAT
        when 0 then 'N'
        else 'Y'
       end as M_LAT,
       IMP.M_HDG_SOLD,
       FLOW.M_VAL_DATE as M_NFAMOUNTD,
       FLOW.M_AMOUNT as M_NFAMOUNT,
       FLOW.M_CURRENCY,
       case FLOW.M_INTERNAL
         when 0 then CTP.M_DSP_LABEL
         else PTF2.M_DSP_LABEL
       end as M_DEST,
       PTF1.M_DSP_LABEL as M_ORIGIN
       from
       TRN_DD_DBF DD, TRN_EXT_DBF EXT, EVT_EVENT_DBF EVT,
       FCE_FLOW_DBF FLOW, TRN_PFLD_DBF PTF2, TRN_CPDF_DBF CTP, TRN_PFLD_DBF PTF1,      EVT_IMP_DBF IMP
       where
       DD.M_LIMPEXTREF *= EXT.M_REFERENCE
       and EXT.M_EVT_REF *= EVT.M_REFERENCE
       and EXT.M_EVT_VS *= EVT.M_VERSION
       and EVT.M_REFERENCE *= FLOW.M_EVT
       and EXT.M_TRADE_REF *= FLOW.M_COMPONENT
       and FLOW.M_DEST *= PTF2.M_LABEL
       and FLOW.M_DEST *= CTP.M_LABEL
       and FLOW.M_SOURCE *= PTF1.M_LABEL
       and DD.M_LIMPREF *= IMP.M_REFERENCE
```

File example 8 : trn_hdrf_vw_dbf.txt

```
create view TRN_HDRF_VW_DBF as
             select M_NB, M_DATE, M_CURRENCY, M_AMOUNT, M_TYPE1 as M_FLOW_TPL1
             from TRN_HDRF_DBF A, FLOW_TYPO_DBF B
             where A.M_FLOW_TYPID = B.M_REF
             union all
             select M_COMPONENT, M_VAL_DATE, M_CURRENCY, M_AMOUNT, M_TYPE1 as M_FLOW_TPL1
             from FCE_FLOW_DBF A, FLOW_TYPO_DBF B
             where A.M_TYPOLOGY = B.M_REF and M_CVERSION = 0
```

File example 9 : process_ps_date_vw_dbf.txt

```
  create view PROCESS#PS_DATE_VW_DBF
  as
       select max(TIMESTAMP) as TIMESTAMP, max(M_IDENTITY) as M_IDENTITY, max(M_DATE_ACC) as M_DATE_ACC, m
> ax(M_DATE_PL) as M_DATE_PL,                          max(M_DATE_FO) as M_DATE_FO
       from
       (
               select max(TIMESTAMP) as TIMESTAMP, max(M_IDENTITY) as M_IDENTITY,max(M_ACC_DATE) as M_DATE_ACC, n
> ull as M_DATE_PL, null as M_DATE_FO          from TRN_ENTD_DBF
               union all
               select null, null, null, max(M_DATE) , null from TRN_PC_DBF
               union all
               select null, null, null, null, max(M_DATE)? from TRN_DSKD_DBF
```