MERVA ESA Components

Traffic Reconciliation

Version 4 Release 1

SH12-6392-00

MERVA ESA Components

Traffic Reconciliation

Version 4 Release 1

SH12-6392-00

Note!

Before using this information and the product it supports, be sure to read the general information under “Appendix D. Notices” on page 147.

First Edition, March 2000

This edition applies to Version 4 Release 1 of IBM MERVA Components (5648-B30) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About This Book

This book describes the Traffic Reconciliation component of Message Entry and Routing with Interfaces to Various Applications MERVA ESA (hereafter referred to as MERVA ESA or MERVA).

Summary of Changes

Version 4 of MERVA for ESA Traffic Reconciliation supports all the functions available in Version 3, plus many new functions and other enhancements.

Functional Enhancements

- All monitors can insert events directly into the DB2 tables bypassing the intermediate "flip-flop" event data set.
- All monitors can alternatively write events to the intermediate event data set (implemented by V3.3 PTF UQ14845 and UQ14846).
- MQI monitor. MQSeries connections using the MERVA-MQI attachment can now be monitored.
- A delete utility is provided to allow old data to be removed from the DB2 database. Deleted database records can optionally be written to a data set.
- A Financial Message Capture transaction allows S.W.I.F.T. messages at any point in the MERVA routing scheme to be inserted to the Traffic Reconciliation SWIFT Link tables. This allows S.W.I.F.T. messages that enter and leave MERVA ESA other than by MERVA SWIFT Link, for example by MERVA-to-MERVA Financial Message Transfer/ESA or MERVA Link, to be recorded by Traffic Reconciliation.
- Users can force the message TRN to be stored in column U_MUR in the S.W.I.F.T. FINxHDR tables. Previously, the TRN would only be stored if the message contained no MUR.
- Fields can be extracted from block 3 of S.W.I.F.T. messages as well as from block 4 (V3.3 PTF UQ23624, UQ23626).
- S.W.I.F.T. messages wrapped in an x9x envelope by a **G9x** or a **I9x** command are now protected from modification.
- Queues to be monitored can be specified using wildcard characters. For example, **L1***: all L1 queues, or ***AI0**: all message authorization queues.
- A scanner control table can be specified for each queue event. Previously only one scanner control table could be specified for all queue monitor events.
- Insertion from the event data sets:
  - A batch version of the event insertion transaction, IMRINSP, is now provided.
  - The event insertion transaction (and the batch version) now switches between flip-flop data sets at end-of-file. Previously it terminated.
  - The nucleus program IMRNUCP, the insertion transaction controller, has been renamed IMRICON to better convey its purpose. It now accumulates insertion counts and displays them at termination. Unless IMRICON is inactive, the insertion transaction no longer displays counts at each transaction termination.
  - IMRICON now initiates the IMRI transaction immediately on starting up (unless MERVA is not yet ready), rather than first waiting for the TIV interval to expire.

  - IMRICON will not re-initiate the transaction until the transaction terminates successfully.
  - If MERVA cannot initiate the transaction, an explanatory message is now issued.
- QRY panel PF-key settings have been changed to accord more with MERVA ESA conventions:
  - A query can be selected by cursor either with the **Enter** key or **PF4**. **PF6** is no longer used.
  - Having selected a query it can now be run with **PF3**, not **PF4**.
  - A selected query can now be escaped from with **PF12**, not **PF3**.
  - A query can no longer be run straight from the QRY selection panel using a PF key. It must first be selected, and then run with **PF3**.
  - When a query is selected from the Query Selection panel, it is initially displayed in standard (PROMPT) format. From that panel, you can display the query in SQL (NOPROMPT) format by pressing **PF11**. Previously, queries were sometimes initially displayed in PROMPT format and sometimes in NOPROMPT format, depending on the format last chosen.
  - The sample access control table for MLINK events (IMRMLKT) inserts the fields extracted by the sample MCB IMRMLKC into the DB2 tables individually, rather than as a single concatenated string.
- Timestamps are now precise to a millionth of a second (partly implemented by V3.3 PTF UQ21616, UQ22443).
- All Traffic Reconciliation modules now run above the 16 MB boundary.
- MERVA ESA online help now also includes Traffic Reconciliation operator messages.

Customization Enhancements

Customization has been simplified. A number of changes have been made to aid understanding:

- All DB2 definitions use default values. The intention is to allow new users to install a first, test, system with a minimum of customization of the DDL.
- Foreign keys (referential integrity) are provided as comments in the DDL (however, their use is not recommended).
- Plan names are no longer hard-coded, alternative names can be specified.
- **SET CURRENT SQLID** has been removed from IMRSQLP, the background processor for MCB-based queries. The program’s plan can now be bound with **DYNAMICRULES(BIND)**.
- The DB2 language interface module is now always loaded dynamically. Post-installation link editing is no longer required.
- IMRPRM parameters module generation.

  The following changes have been made to macro IMRPARM:
  - Parameter checking has been improved.
  - The separate macros for each monitor have been replaced by macro IMRPARM with **TYPE=EVENT**.
  - The **QNAME** parameter, part of the resource name for MVS resource serialization, is no longer used. The **NAME** parameter from the MERVA ESA parameters module, DSLPRM, is used instead.
  - Use of the **USER** parameter is now an error. A MERVA Link exit must be specified in MERVA Link macro EKAPT.

  - Parameter **TLXFLDS** replaces **FORMID** and **MSGID** in the Telex Link monitor specification. Specification of formatting for field extraction is thus more consistent across all monitors.
  - When specifying a MERVA message identifier, the more correct keyword **MTYPE** is used instead of “**MCB**”.
  - To specify a scanner control table, the parameter **CONTROL** has been renamed to the more precise **SCANTAB**.
  - The name of the parameter for the flip-flop threshold value has been corrected to **FFTHRESH**.
  - The **APPC** parameter for MERVA Link monitors is no longer used.
- Scanner control tables:
  - The macros IMRQUS and IMRTXS have been combined and renamed IMRSCAN.
  - **BEG** and **END** tag specifications can combine hexadecimal and character notations. For example, instead of coding:

    ```
    BEG=X'0D257A6F6F6F7A',END=X'0D2560'
    ```

    you can now code:

    ```
    BEG=(X'0D25',C':???:'),END=(X'0D25',C'-')
    ```
  - The **LIN** parameter has been renamed **LINE**.
- Table filter, macro IMRTFL:
  - The **TYPE=ENTRY** parameter has been renamed **TYPE=TABLE**.
  - The IMRFLD macro has been integrated into macro IMRTFL and is invoked with **TYPE=FIELD**.

Note: With few exceptions V3 customization tables can continue to be used in V4 without change.

DB2 Tables

- S.W.I.F.T. tables containing an MIR or MOR have been expanded to contain the subfields of the MIR or MOR as separate columns. This allows these subfields to be indexed.
- Views have been defined to provide easier access to the S.W.I.F.T. LT as a whole in all S.W.I.F.T. tables containing a logical terminal.
- A new column, EKEY, has been added to the MERVA Link and Telex Link HDR tables (V3 PTF UQ14845, UQ14846).
- The spelling of column MSG_CATEGORIE in the MERVA Link HDR tables has been corrected to MSG_CATEGORY.
- Indexes are no longer defined on subordinate table primary keys.

Others

- S.W.I.F.T. incoming GPA NAKs were being stored in table GPAOSER instead of GPAISER. This has been corrected.
- All batch programs pass a return code to the operating system.
- The queue batch utility (IMRQMGB) has been moved to the MERVA ESA base product and renamed DSLSQB. See the *MERVA for ESA V4 Operations Guide*.

Part 1. Introducing Traffic Reconciliation

Chapter 1. Overview

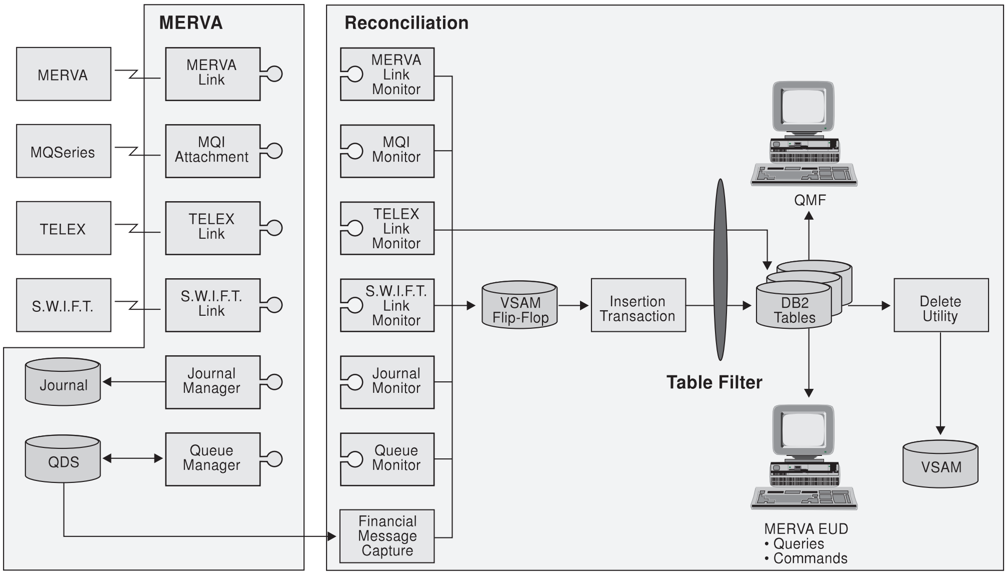


Figure 1. MERVA and Traffic Reconciliation Overview

Traffic Reconciliation is a MERVA component that captures and stores in DB2® tables data about MERVA ESA activity, for example:

- The movement of messages across networks
- The movement of messages within the MERVA ESA system
- The content of messages

The scope and detail of the data extracted can be extensively customized.

MERVA ESA end users can use Traffic Reconciliation to monitor the functioning of their messaging applications and as a message archive, managers can use it to gather statistics about message traffic, and the MERVA ESA administrator can use it to monitor the functioning of MERVA ESA.

Monitors

There are six components of MERVA ESA from which data can be extracted by Traffic Reconciliation:

- SWIFT Link
- Telex Link
- MERVA Link
- MERVA-MQI Attachment
- MERVA journal
- MERVA queue manager

Traffic Reconciliation defines a monitor for each of these components that performs the component-specific data extraction. For each monitor, a set of DB2 tables are defined into which the data is stored.

The following network monitors record messages that are sent and received over their respective networks:

- SWIFT Link
- Telex Link
- MERVA Link
- MERVA-MQI attachment

S.W.I.F.T. messages that enter and leave the MERVA system via an application other than SWIFT Link (for example, via MERVA Link) can nevertheless be recorded in the SWIFT Link tables by the Financial Message Capture transaction.

The MERVA journal and queue manager monitors record internal MERVA ESA activity. The queue manager monitor can record any movement of a message in the MERVA message queuing system. The journal monitor can record any record written to the MERVA journal.

The central customization module for reconciliation (IMRPRM) defines which monitors are active.

SWIFT Link monitor

The SWIFT Link monitor, program IMRSWFX, is a MERVA central service and is invoked by the SWIFT Link modules DWSDGPAS (sending) and DWSDGPAR (receiving). IMRSWFX is linked to the MERVA nucleus.

Telex Link monitor

The Telex Link monitor records MERVA ESA Telex Link events. The Telex Link monitor program (IMRTLXP) is a load module loaded and invoked by the Telex Link substation interface program ENLHCF1.

Queue monitor

The queue monitor captures MERVA queue management events. A queue management event is either the storing of a message into a MERVA queue, or the deletion of a message from a MERVA queue. For example, if a message is routed from queue A to queue B, then two events are generated: a delete event from queue A, and a put event to queue B..

The queue mangement monitor program, IMRQUEX, is a MERVA central service and is invoked by the queue management programs DSLQMGT (VSAM QDS) and DSLQMGD (DB2). IMRQUEX is linked to the MERVA nucleus.

Notes:

1. The following queue management events are not recorded:
   - Replace (API “REPL” function)
   - Storing to a dummy queue
2. For the following events the UMR is a string of all zeros:
   - The test commands DELETE and MOVE
   - API “DELE” function

Journal Monitor

The journal monitor captures MERVA journal events, the writing of journal records. Only the journal record is recorded, fields cannot be extracted from the record.

Note: Segmented journal records are not recorded.

The journal monitor, IMRJRNX, is a MERVA central service and is invoked by DSLJRNP. IMRJRNX is linked to the MERVA nucleus.

MERVA Link Monitor

The MERVA Link monitor monitors one or more MERVA Link ASPs, which are defined in the MERVA Link partner table EKAPT. The MERVA Link monitor program (IMRMLKP) is an MFS user exit invoked by the MERVA Link sending ASP (EKAAS10) and receiving ASP (EKAAR10) immediately after the exit EKAPT MFSEXIT is invoked.

MERVA-MQI Attachment Monitor

The MQI monitor records messages transferred between MERVA and MQSeries by one or more MERVA-MQI attachment processes, which are defined in the MERVA-MQI attachment process table DSLKPROC. The MQI monitor program (IMRMQIP) is an MFS user exit invoked by the MERVA-MQI attachment send and receive transactions DSLKQS and DSLKQR.

Events

A unit of data extracted from MERVA is termed an “event”. Depending on its type, an event can contain MERVA-specific data, an application message in network format, and other mappings of the message for the purpose of extraction of individual message fields.

The parameter module IMRPRM also defines which events are to be extracted, and how messages are to be mapped.

Messages

For proper documentation of MERVA ESA external message interface events (S.W.I.F.T., Telex, MERVA Link, MQI) the message as transmitted across the network is recorded. For MERVA Link, which can transfer messages in MERVA ESA queue format as well as in net format, you must specify how Traffic Reconciliation is to format this message documentation string.

For Queue and Journal events a message string is not recorded.

Message Fields

In addition to the message in net format, individual fields in the message can be extracted for insertion into DB2 tables. These fields are then available to qualify database queries.

Fields are extracted by scanning a special message string for specific tags and inserting any substrings thus identified into field tables. You must specify how Traffic Reconciliation is to format this message string for field extraction, and how each extracted field is to be identified in the field tables. Message formatting is specified in the parameters module, IMRPRM, field extraction is specified in a scanner control table.

A message string for field extraction is not required for S.W.I.F.T. messages since they already have well-known message formats and tags. Fields are extracted from the network format of S.W.I.F.T. messages using the standard S.W.I.F.T. field tags.

For some monitors, field extraction can be restricted to fields in specific message types by adding field specifications to the Traffic Reconciliation table filter.

You cannot extract fields from journal events.

Field extraction is summarized in the following table:

Table 1. Message Field Extraction

Monitor	Field Extraction	Field String Formatting	Uses Scanner Control Table	Qualified by Table Filter
S.W.I.F.T.	Yes	No	No	Yes
MERVA Link	Yes	Yes	Yes	Yes
MQI	Yes	Yes	Yes	Yes
Telex	Yes	Yes	Yes	No
Queue	Yes	Yes	Yes	No
Journal	No	No	No	No

The Flip-Flop

Event data captured by the Traffic Reconciliation monitors can be either inserted directly into the DB2 tables, or written to the flip-flop for asynchronous insertion into DB2 by the insertion transaction IMRI.

The flip-flop is a pair of data sets that are filled alternately. When one data set becomes full, the extraction process (the event capture process) switches to the other data set. Simultaneously, the insertion transaction can be reading event records from the flip-flop and inserting data into the DB2 database. When all records from one data set have been processed by the insertion transaction, that data set is made available again to the extraction process. The two data sets have ddnames IMRDSA and IMRDSB.

The flip-flop is in effect an event buffer, allowing MERVA ESA and the extraction process to be insulated from DB2 and the insertion process.

CTL Data Set

A third data set, the control data set, contains cursors into the flip-flop data sets that control writing and reading of events. Note that at MERVA termination, although the insertion transaction is initiated one last time, events can still be inserted into the flip-flop after the transaction completes, in particular by MERVA Link and MERVA-MQI attachment receive transactions. Consequently, it can happen that these last events are not inserted into the DB2 tables.

Tables

Traffic Reconciliation defines over 70 DB2 tables:

29	S.W.I.F.T. FIN
21	S.W.I.F.T. GPA
8	MERVA Link
7	Telex Link
10	MQI
4	Queue Manager
1	Journal

All tables have a name in the form **aaaxttt** or **aaattt**, where:

aaa The application (FIN, GPA, TLX, MLK, MQI, QUE, JRN)

x The letter **I** or **O**, indicating the direction of the movement of the application message (I for input, O for output); this does not apply to queue tables or the journal table

ttt The type of the table, for example HDR, DOC, or FLD

"Appendix A. DB2 Table Overview" on page 95 shows an overview of all Traffic Reconciliation tables.

Input and Output

Events are recorded in different sets of tables depending on the direction of the movement of the application message. The terms *input* and *output* are used by Traffic Reconciliation in the S.W.I.F.T. sense:

- *Input* for messages going from MERVA to a network
- *Output* for messages entering MERVA from a network

The concept of input and output has no meaning for journal and queue events.

Types of Tables

Tables can be of different types:

HDR Tables
Of the tables that reflect a particular event, the header table is the central table.

DOC Tables
The application message, as transferred across the network, is always recorded as a string in the appropriate documentation table.

FLD Tables
There are a number of tables into which individual fields from an application message can be inserted. Field tables contain a column to identify the field (usually **FIELD_ID**) and a column for the fields contents (usually **FIELD_DATA**).

Fields from S.W.I.F.T. events can be inserted into amount tables (xxxyAMN, xxxyF32), reference field tables (xxxyREF), and general field tables (xxxyFLD).

Joining Tables

Some of the columns in the tables can be used to join tables:

PKEY Primary key. All tables contain a PKEY column, which contains each record's unique identifier.

DKEY Documentation key. This is the PKEY of a DOC table.

AKEY Associated key. This is the PKEY of a related HDR or ACK table.

UMR The MERVA unique message reference, which lets different events concerning a particular message be joined across monitor boundaries, for example S.W.I.F.T. and queue events.

The EKEY column is not a true key. It contains the event timestamp, that is, the time the event occurred in MERVA.

Queries

A number of SQL queries are supplied with Traffic Reconciliation. These queries can be used to extract information from the DB2 database, and can serve as examples on which users can base their own queries.

Queries can be of two types:

- **MCB-based queries**, which are queries integrated into the MERVA ESA end user terminal environment (see “Issuing MCB-Based Queries” on page 33)
- **QMF™ queries**, which can be either invoked from a TSO session or run as a batch program (see “Issuing QMF Queries” on page 38)

Note: Some queries reference field tables (FLD, REF, AMN, and others) in the expectation that field extraction has been defined as specified in the distributed scanner control tables and table filter.

Chapter 2. DB2 Tables Used by Traffic Reconciliation

This chapter describes the DB2 tables used by Traffic Reconciliation. See “Appendix A. DB2 Table Overview” on page 95 for an overview of all Traffic Reconciliation tables.

SWIFT Link Monitor Tables

The S.W.I.F.T. monitor tables are grouped by S.W.I.F.T. application (FIN and GPA), and by the direction of message flow (input and output). The group to which a table belongs is indicated by the first four letters of its name:

FINIttt	FIN input APDUs[1] and ISN ACKs
FINOttt	FIN output APDUs and OSN ACKs
GPAIttt	GPA input APDUs and ISN ACKs
GPAOttt	GPA output APDUs and OSN ACKs

The **ttt** represents a three-letter code indicating the table’s type:

- Documentation table (DOC)

  The table records messages (APDUs) in the original S.W.I.F.T. format.
- Header table for service, system, and user-to-user messages (HDR)

  The table contains information extracted from the basic headers, application headers, and user headers of S.W.I.F.T. messages.
- Header table for ISN/OSN ACKs (ACK)

  The table contains information extracted from APDU 21 messages, that is, messages with service identifier 21.

  Note: Acknowledgments to messages (APDU 21) are stored together with the corresponding acknowledged message; ISN ACKs are stored in the input table set, and OSN ACKs in the output tables.
- Trailer table (TRL)

  The table contains information extracted from the S.W.I.F.T. trailer block.
- Text block tables (AMN, F32, FLD, MIR, MOR, MUR, REF, SEC, SER, SES)

  These tables contain information extracted from the user header and text block of the S.W.I.F.T. messages (blocks 3 and 4).
- MERVA ESA related tables (SRC, TAR, MER)

  These tables contain data about the sending queue and the target queues from MERVA ESA input, output, or error routing during the process of sending or receiving S.W.I.F.T. messages.

Central to each set of tables are the header tables, which are related to the MERVA ESA related tables by the UMR and to the documentation tables by the documentation key (DKEY). The relation to the text block tables and the trailer table is by the AKEY column. The relation between the header and the ACK table is by the S.W.I.F.T. basic header.

[1] The term **APDU** is used in Traffic Reconciliation to refer to a message string as transferred across a network. A **PDU** (protocol data unit) is a unit of data defined in a communication protocol. An APDU is a PDU that contains an application message, for example, a S.W.I.F.T. message. **APDU 21** refers to a S.W.I.F.T. message with service identifier 21.

Depending on the specifications in the Traffic Reconciliation parameters module IMRPRM and the table filter, events may not be fully recorded in the tables.

Insertion can be restricted to particular groups of messages (user-to-user, system, service) by the table filter **SCOPE** parameter.

Figures 2 to 5 show the S.W.I.F.T. tables and how they are related to each other.

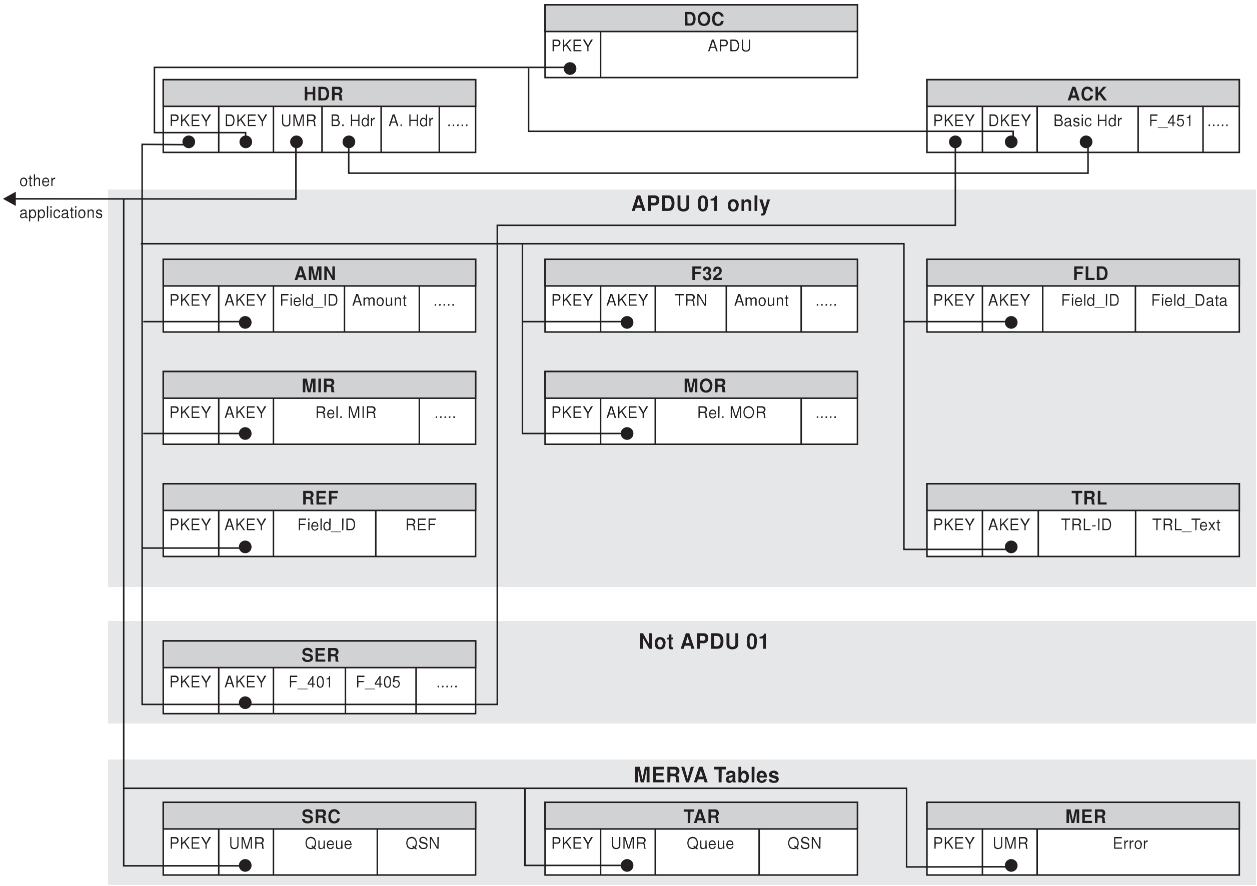


Figure 2. FIN Input Tables

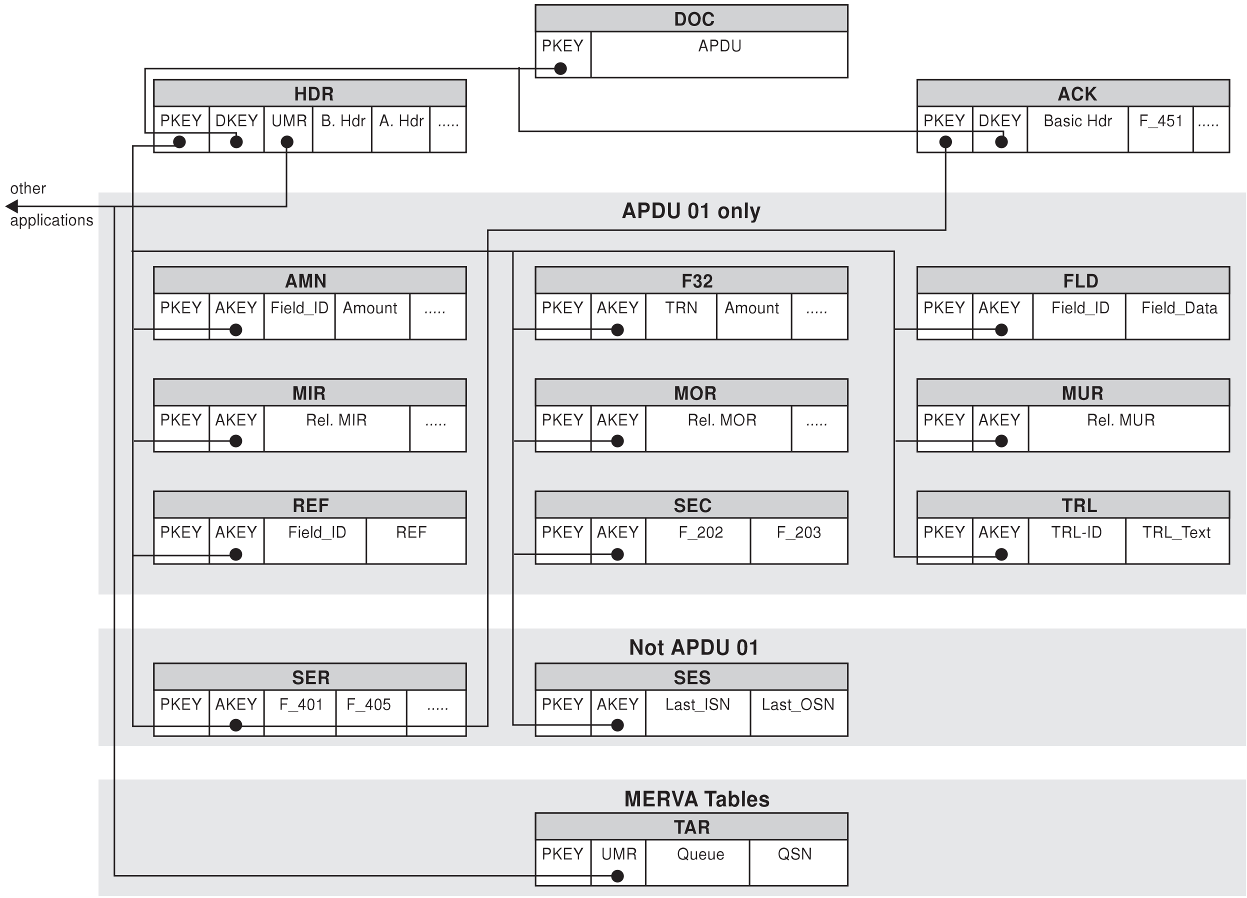


Figure 3. FIN Output Tables

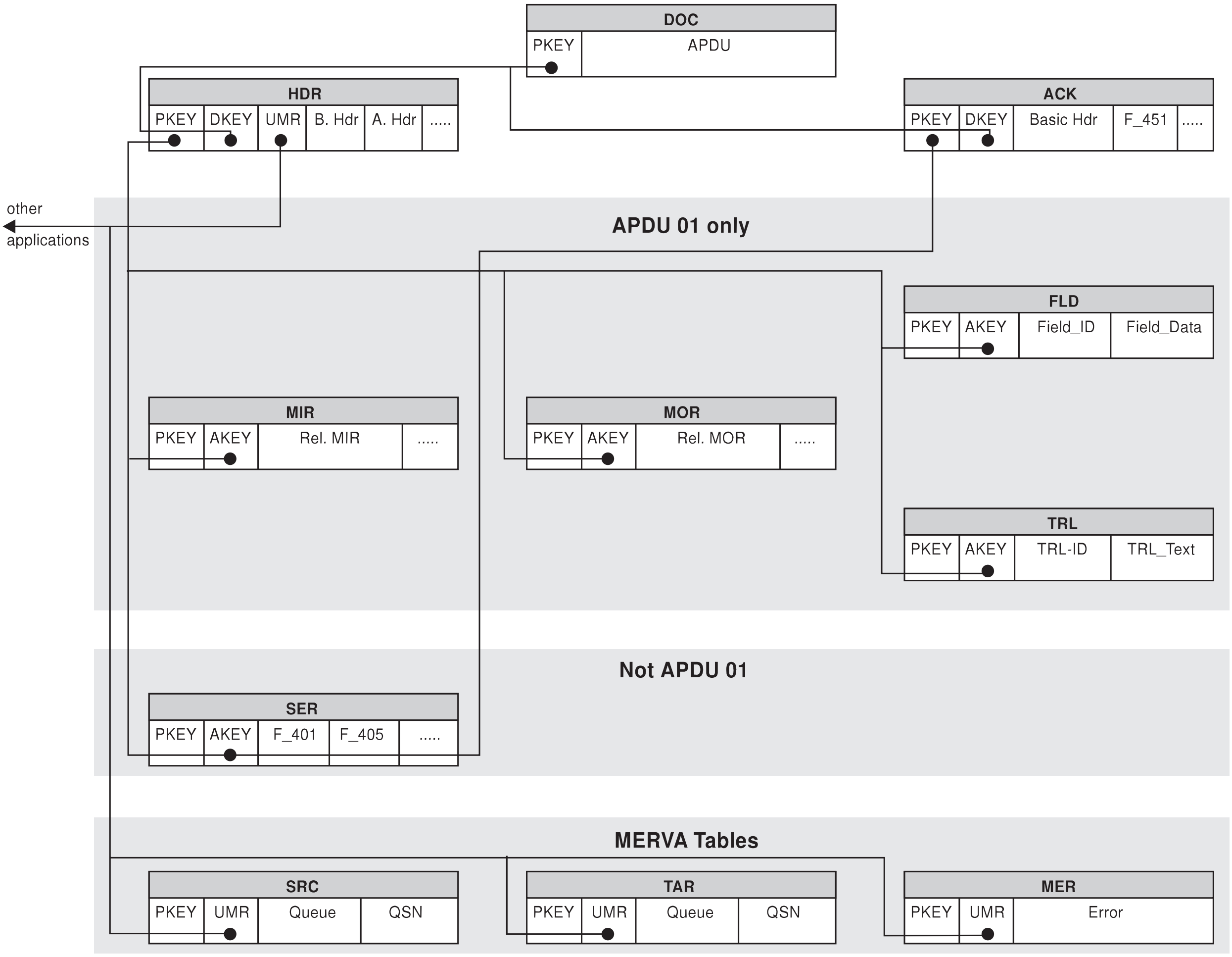


Figure 4. GPA Input Tables

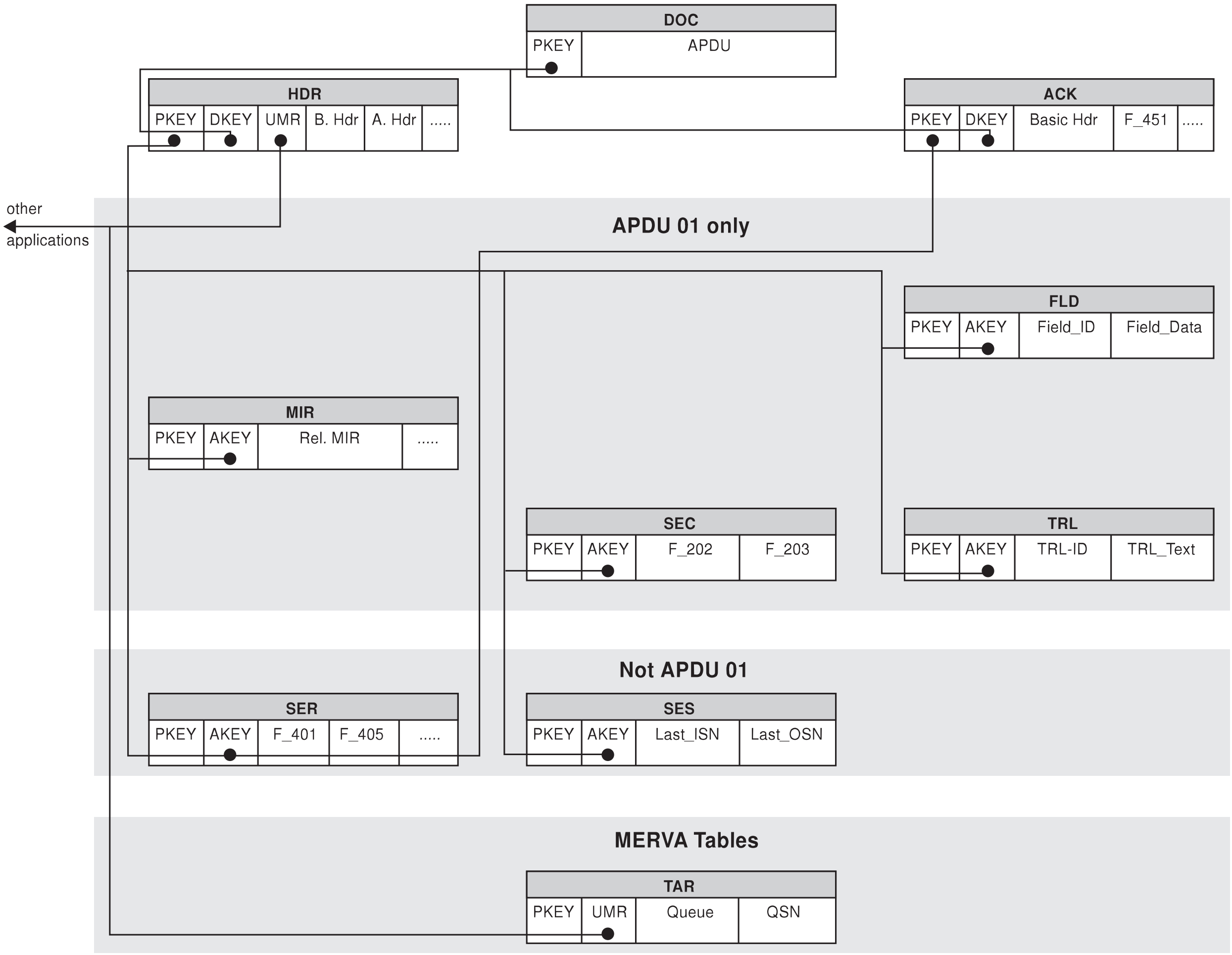


Figure 5. GPA Output Tables

DOC Tables

The DOC tables (FINIDOC, FINODOC, GPAIDOC, GPAODOC) contain S.W.I.F.T. APDUs and the corresponding ACKs and NAKs. In the set of SWIFT Link tables the DOC table is the root table.

You can use the **SCOPE** parameter in the table filter entry for a DOC table (described on page 63) to inhibit storage of particular categories of S.W.I.F.T. messages in the database. These categories are:

- S.W.I.F.T. service messages
- S.W.I.F.T. system messages
- User-to-user messages
- Acknowledgments

PKEY Timestamp of insertion. This is the primary key and is identical to the DKEY in the header or ACK tables.

APDU S.W.I.F.T. APDU in its original format.

DOC tables have definitions of the form:

```
CREATE TABLE FINIDOC
  ( PKEY            TIMESTAMP NOT NULL      -- insertion timestamp
   ,APDU            VARCHAR(11000) NOT NULL -- msg.in net format
   ,PRIMARY KEY (PKEY)
  )
```

HDR Tables

The HDR tables (FINIHDR, FINOHDR, GPAIHDR, GPAOHDR) contain information from the message basic and application headers, but not from ISN or OSN ACKs.

PKEY Timestamp of insertion. This is the primary key and is identical to the AKEY in the text block/trailer tables.

EKEY Timestamp of the occurrence of the event.

DKEY PKEY of this APDU in the documentation table.

APDU_LENGTH
Length of the APDU in the DOC table.

LINE_NUMBER
Not used.

RETRIEVAL Indicates whether the information associated with this APDU belongs to a message extracted from a retrieval. For every MT 021 the embedded message is also processed separately. This message is marked with a **Y** in this column. Otherwise the column contains a blank.

MAST_DEST Master destination according to the MERVA ESA logical terminal table (DWSLTT).

MUR_TRN Indicates the contents of the column U_MUR:

M MUR

T TRN

(blank) nothing

By default, the MUR is inserted into the column U_MUR; if no MUR is found, the program searches for the first TRN (field 20) in the message.

You can force the U_MUR column always to contain the TRN by specifying **TRNINHDR=YES** in the IMRPRM **TYPE=EVENT** specification (see page 60). You can still record the MUR as well by selecting it for insertion into an FLD or REF table. You do this by adding a **FIELDS=108** specification to the table filter entry for the required table.

U_MUR The user message reference field contains either the S.W.I.F.T. MUR or the first TRN of the message. The distinction is made by the MUR_TRN indicator.

The other columns are from the S.W.I.F.T. basic and application headers and are self-explanatory.

HDR tables have definitions of the form:

```
CREATE TABLE FINIHDR
  ( PKEY            TIMESTAMP NOT NULL     -- insertion timestamp
   ,EKEY            TIMESTAMP NOT NULL     -- event time
   ,DKEY            TIMESTAMP NOT NULL     -- DOC table PKEY
   ,UMR             CHAR(28)  NOT NULL     -- MERVA Unique Msg.Ref.no.
   ,APDU_LENGTH     SMALLINT  NOT NULL     -- length of DOC.APDU
   ,LINE_NUMBER     SMALLINT  NOT NULL     -- (not used)
   ,RETRIEVAL       CHAR(1)   NOT NULL     -- Y: extracted from retr.
   ,MAST_DESTIN     CHAR(8)   NOT NULL     -- master destination (DWSLTT)
   ,MUR_TRN         CHAR(1)   NOT NULL     -- M/T: col U_MUR = MUR/TRN
   ,B_APPLID        CHAR(1)   NOT NULL     -- F (FIN)
   ,B_APDUID        CHAR(2)   NOT NULL     -- SWIFT service identifier
   ,B_BANK          CHAR(4)   NOT NULL     -- senders LT address..
   ,B_COUNTRY       CHAR(2)   NOT NULL     --
   ,B_LOCATION      CHAR(2)   NOT NULL     --
   ,B_TERMINAL      CHAR(1)   NOT NULL     --
   ,B_BRANCH        CHAR(3)   NOT NULL     --
   ,B_SESSION       CHAR(4)   NOT NULL     -- session number
   ,B_SEQUENCE      CHAR(6)   NOT NULL     -- ISN
   ,A_MSGTYPE       CHAR(3)   NOT NULL     -- MT number
   ,A_DESTIN        CHAR(8)   NOT NULL     -- receivers LT
   ,A_BRANCH        CHAR(3)   NOT NULL     -- receivers branch code
   ,A_PRIORITY      CHAR(1)   NOT NULL     -- S/U/N
   ,A_DELIVERY      CHAR(1)   NOT NULL     -- 1/2/3
   ,A_OBSOLES       CHAR(3)   NOT NULL     -- obsolescence period
   ,U_MUR           CHAR(16)  NOT NULL     -- MUR or TRN
   ,PRIMARY KEY (PKEY)
-- ,FOREIGN KEY (DKEY) REFERENCES FINIDOC ON DELETE CASCADE
  )
```

Views, FINIHDR2 and FINOHDR2, are defined on the corresponding tables to allow the logical terminal columns in the basic header to be referenced as a single column, B_LT.

ACK Tables

The ACK tables (FINIACK, FINOACK, GPAIACK, GPAOACK) contain header information for ISN and OSN ACKs.

F_451 Field 451: It indicates acceptance ('0') or rejection ('1') of the message by S.W.I.F.T.

ACK_DATE Date from field 177

ACK_TIME	Time from field 177

ACK tables have definitions of the form:

```
CREATE TABLE FINIACK
  ( PKEY            TIMESTAMP NOT NULL     -- insertion timestamp
   ,EKEY            TIMESTAMP NOT NULL     -- event time
   ,DKEY            TIMESTAMP NOT NULL     -- DOC table PKEY
   ,B_BANK          CHAR(4)   NOT NULL     -- LT of acknowledged msg
   ,B_COUNTRY       CHAR(2)   NOT NULL     --
   ,B_LOCATION      CHAR(2)   NOT NULL     --
   ,B_TERMINAL      CHAR(1)   NOT NULL     --
   ,B_BRANCH        CHAR(3)   NOT NULL     --
   ,B_SESSION       CHAR(4)   NOT NULL     -- ..session number
   ,B_SEQUENCE      CHAR(6)   NOT NULL     -- ..ISN
   ,ACK_DATE        CHAR(6)   NOT NULL
   ,ACK_TIME        CHAR(4)   NOT NULL
   ,F_451           CHAR(1)   NOT NULL     -- 0:accepted, 1: rejected
   ,PRIMARY KEY (PKEY)
-- ,FOREIGN KEY (DKEY) REFERENCES FINIDOC ON DELETE CASCADE
  )
```

Views are defined on the ACK tables to allow the LT in the basic header to be referenced as a single column, B_LT.

System and User-to-User Message Tables (APDU 01)

AMN Tables

The AMN tables (FINIAMN, FINOAMN) list the tags and component parts of amount fields found in the text block (block 4) of S.W.I.F.T. messages. Specifications in the table filter can limit the fields selected.

An amount field is any field having the following structure:

1. An optional 6-digit date, followed by
2. A 3-character currency code, followed by
3. A varying number of digits followed by a comma and, optionally,
4. A further sequence of digits

The columns contain:

FIELD_ID	Amount field tag with option
AMOUNT	Integer part of the amount
AMOUNTD	Fractional part of the amount, in 10,000ths
DATE	Value date
CURRENCY	Currency code

AMN tables have definitions of the form:

```
CREATE TABLE FINIAMN
  ( PKEY            TIMESTAMP NOT NULL     -- insertion timestamp
   ,AKEY            TIMESTAMP NOT NULL     -- HDR table PKEY
   ,FIELD_ID        CHAR(3)   NOT NULL     -- SWIFT field tag
   ,AMOUNT          DEC(15,0) NOT NULL     -- before the comma
   ,AMOUNTD         DEC(4,0)  NOT NULL     -- after the comma
   ,CURRENCY        CHAR(3)   NOT NULL     -- currency code
   ,DATE            CHAR(6)   NOT NULL     -- value date
-- ,FOREIGN KEY (AKEY) REFERENCES FINIHDR ON DELETE CASCADE
  )
```

F32 Tables

The F32 tables (FINIF32, FINOF32) record TRN, date, and amount groups from the following messages:

MT100	Field 20, 32
MT200	Field 20, 32
MT201	Field 30, 20, 32 (multiple rows)
MT202	Field 20, 32
MT203	Field 30, 20, 32 (multiple rows)
MT205	Field 20, 32

In contrast, the FINxAMN tables can record amount fields regardless of message type.

The columns are:
TRN Transaction number of the amount group
AMOUNT Integer part of the amount
AMOUNTD Fractional part of the amount, in 10,000ths
DATE Value date

F32 tables have definitions of the form:

```
CREATE TABLE FINIF32
  ( PKEY            TIMESTAMP NOT NULL     -- insertion timestamp
   ,AKEY            TIMESTAMP NOT NULL     -- HDR table PKEY
   ,TRN             CHAR(16)  NOT NULL
   ,AMOUNT          DEC(15,0) NOT NULL     -- before the comma
   ,AMOUNTD         DEC(4,0)  NOT NULL     -- after the comma
   ,CURRENCY        CHAR(3)   NOT NULL     -- currency code
   ,DATE            CHAR(6)   NOT NULL     -- value date
-- ,FOREIGN KEY (AKEY) REFERENCES FINIHDR ON DELETE CASCADE
  )
```

FLD Tables

The FLD tables (FINIFLD, FINOFLD, GPAIFLD, GPAOFLD) record field tags or subblock identifiers and field data of any fields in the user header (block 3) and text block (block 4).

The columns contain:

FIELD_ID Field tag with option or subblock identification (for example, 32A,107)

FIELD_DATA Data areas or subblock data of the field

FLD tables have definitions of the form:

```
CREATE TABLE FINIFLD
  ( PKEY            TIMESTAMP NOT NULL      -- insertion timestamp
   ,AKEY            TIMESTAMP NOT NULL      -- HDR table PKEY
   ,FIELD_ID        CHAR(3)   NOT NULL      -- SWIFT field tag
   ,FIELD_DATA      VARCHAR(   69) NOT NULL -- field value
-- ,FOREIGN KEY (AKEY) REFERENCES FINIHDR ON DELETE CASCADE
  )
```

MIR Tables

The MIR tables (FINIMIR, FINOMIR, GPAIMIR, GPAOMIR) record, in the column RELATED_MIR, the related MIRs from the following messages:

MT010	Field 106
MT011	Field 106
MT015	SYS trailer

MT019	Field 106
MT020	Field 251 or 252 (two rows)
MT022	Field 251 or 252 (two rows)
MT024	Field 106
MT039	Field 106
MT059	Field 106
MT066	Field 335
MT082	Field 335
MT083	Field 335

MIR tables have definitions of the form:

```
CREATE TABLE FINIMIR
  ( PKEY            TIMESTAMP NOT NULL     -- insertion timestamp
   ,AKEY            TIMESTAMP NOT NULL     -- HDR table PKEY
   ,RELATED_MIR     CHAR(28)  NOT NULL     -- MIR
   ,MIR_DATE        CHAR(6)   NOT NULL     -- RELATED_MIR subfields..
   ,MIR_BANK        CHAR(4)   NOT NULL
   ,MIR_COUNTRY     CHAR(2)   NOT NULL
   ,MIR_LOCATION    CHAR(2)   NOT NULL
   ,MIR_TERMINAL    CHAR(1)   NOT NULL
   ,MIR_BRANCH      CHAR(3)   NOT NULL
   ,MIR_SESSION     CHAR(4)   NOT NULL
   ,MIR_SEQUENCE    CHAR(6)   NOT NULL
-- ,FOREIGN KEY (AKEY) REFERENCES FINIHDR ON DELETE CASCADE
  )
```

Views FINIMIR2, FINOMIR2, GPAIMIR2, and GPAOMIR2 are defined to allow the logical terminal columns to be referenced as a single column. For example, FINIMIR2 is defined as follows:

```
CREATE VIEW  FINIMIR2                      -- alternative MIR view
  ( PKEY
   ,AKEY
   ,RELATED_MIR
   ,MIR_DATE
   ,MIR_LT
   ,MIR_SESSION
   ,MIR_SEQUENCE
  ) AS SELECT
    PKEY
   ,AKEY
   ,RELATED_MIR
   ,MIR_DATE
   ,MIR_BANK CONCAT MIR_COUNTRY CONCAT MIR_LOCATION CONCAT
      MIR_TERMINAL CONCAT MIR_BRANCH
   ,MIR_SESSION
   ,MIR_SEQUENCE
  FROM FINIMIR;
```

MOR Tables

The MOR tables (FINIMOR, FINOMOR, GPAIMOR) record, in the column RELATED_MOR, the related MORs from the following messages:

MT011	Field 107
MT019	Field 107
MT020	Field 253 or 254 (two rows)
MT022	Field 253 or 254 (two rows)

MOR tables have definitions of the form:

```
CREATE TABLE FINIMOR
  ( PKEY           TIMESTAMP NOT NULL     -- insertion timestamp
   ,AKEY           TIMESTAMP NOT NULL     -- HDR table PKEY
   ,RELATED_MOR    CHAR(28)  NOT NULL     -- MOR
   ,MOR_DATE       CHAR(6)   NOT NULL     -- RELATED_MOR subfields..
   ,MOR_BANK       CHAR(4)   NOT NULL
   ,MOR_COUNTRY    CHAR(2)   NOT NULL
   ,MOR_LOCATION   CHAR(2)   NOT NULL
   ,MOR_TERMINAL   CHAR(1)   NOT NULL
   ,MOR_BRANCH     CHAR(3)   NOT NULL
   ,MOR_SESSION    CHAR(4)   NOT NULL
   ,MOR_SEQUENCE   CHAR(6)   NOT NULL
-- ,FOREIGN KEY (AKEY) REFERENCES FINIHDR ON DELETE CASCADE
  )
```

Views FINIMOR2, FINOMOR2, and GPAOMOR2 are defined to facilitate access to the logical terminal columns.

MUR Table

The MUR table (FINOMUR) records, in the column RELATED_MUR, the related MURs from the following messages:

MT010	Field 108
MT011	Field 108
MT019	Field 108

FINOMUR is defined as follows:

```
CREATE TABLE FINOMUR
  ( PKEY           TIMESTAMP NOT NULL     -- insertion timestamp
   ,AKEY           TIMESTAMP NOT NULL     -- HDR table PKEY
   ,RELATED_MUR    CHAR(16)  NOT NULL
-- ,FOREIGN KEY (AKEY) REFERENCES FINOHDR ON DELETE CASCADE
  )
```

REF Tables

The REF tables (FINIREF, FINOREF) hold reference fields from S.W.I.F.T. FIN messages. Any fields in the user header (block 3) and text block (block 4) can be recorded. The REF table would normally be restricted to fields 20 and 21.

The columns contain:

FIELD_ID Field tag with option, or subblock identification

FIELD_DATA Data areas or subblock data of the field

REF tables have definitions of the form:

```
CREATE TABLE FINIREF
  ( PKEY           TIMESTAMP NOT NULL     -- insertion timestamp
   ,AKEY           TIMESTAMP NOT NULL     -- HDR table PKEY
   ,FIELD_ID       CHAR(3)   NOT NULL     -- SWIFT field tag
   ,REFERENCE      CHAR(16)  NOT NULL     -- field value
-- ,FOREIGN KEY (AKEY) REFERENCES FINIHDR ON DELETE CASCADE
  )
```

SEC Tables

The SEC tables (FINOSEC, GPAOSEC) list the section numbers in the S.W.I.F.T. fields 202 and 203 of S.W.I.F.T. output messages.

SEC tables have definitions of the form:

```
CREATE TABLE FINOSEC
  ( PKEY            TIMESTAMP NOT NULL     -- insertion timestamp
   ,AKEY            TIMESTAMP NOT NULL     -- HDR table PKEY
   ,F_202           CHAR(4)   NOT NULL
   ,F_203           CHAR(4)   NOT NULL
-- ,FOREIGN KEY (AKEY) REFERENCES FINOHDR ON DELETE CASCADE
  )
```

TRL Tables

The TRL tables (FINITRL, FINOTRL, GPAITRL, GPAOTRL) contain the S.W.I.F.T. message trailers.

TRL_ID The trailer identifier (PDE, PDM, MAC, and so on)

TRL_TEXT The trailer

Insertion of particular trailers can be inhibited by the **SCOPE** parameter in the table filter, which is described on page 63.

TRL tables have definitions of the form:

```
CREATE TABLE FINITRL
  ( PKEY            TIMESTAMP NOT NULL      -- insertion timestamp
   ,AKEY            TIMESTAMP NOT NULL      -- HDR table PKEY
   ,TRL_ID          CHAR(3)   NOT NULL      -- trailer code
   ,TRL_TEXT        VARCHAR(   32) NOT NULL -- trailer information
-- ,FOREIGN KEY (AKEY) REFERENCES FINIHDR ON DELETE CASCADE
  )
```

Service Message Tables (Not APDU 01)

SER Tables

The SER tables (FINISER, FINOSER, GPAISER, GPAOSER) record S.W.I.F.T. errors related to the indicated S.W.I.F.T. fields for the following APDUs and message types (MTs):

APDU 12	Field 443
APDU 13	Field 441
APDU 14	Field 443
APDU 15	Field 441
APDU 21	Field 405 and 451
APDU 25	Field 401
APDU 26	Field 401
APDU 33	Field 441
APDU 35	Field 441
APDU 42	Field 503
APDU 43	Field 503
MT010	Field 431
MT015	Field 405
MT019	Field 432
MT021	Field 421 and 431
MT023	Field 421 and 431
MT059	Field 442 and 451

SER tables have definitions of the form:

```
CREATE TABLE FINISER
  ( PKEY            TIMESTAMP NOT NULL     -- insertion timestamp
   ,AKEY            TIMESTAMP NOT NULL     -- HDR/ACK table PKEY
   ,F_401           CHAR(2)   NOT NULL
   ,F_405           CHAR(3)   NOT NULL
   ,F_421           CHAR(3)   NOT NULL
   ,F_431           CHAR(2)   NOT NULL
   ,F_432           CHAR(2)   NOT NULL
   ,F_441           CHAR(3)   NOT NULL
   ,F_442           CHAR(2)   NOT NULL
   ,F_443           CHAR(3)   NOT NULL
   ,F_451           CHAR(1)   NOT NULL
   ,F_461           CHAR(3)   NOT NULL
   ,F_503           CHAR(3)   NOT NULL
-- --,foreign key (AKEY) references FINIHDR or(!) FINIACK
  )
```

Note: Since the table can have either the HDR or ACK table as parent, you cannot define a foreign key.

SES Tables

The SES tables (FINOSES, GPAOSES) record the last ISN and last OSN from S.W.I.F.T. fields 331 and 333 when processing the following S.W.I.F.T. Service APDUs:

APDU 22	Field 333
APDU 23	Field 333
APDU 25	Field 331
APDU 26	Field 331

SES tables have definitions of the form:

```
CREATE TABLE FINOSES
  ( PKEY            TIMESTAMP NOT NULL     -- insertion timestamp
   ,AKEY            TIMESTAMP NOT NULL     -- HDR table PKEY
   ,LAST_ISN        CHAR(6)   NOT NULL
   ,LAST_OSN        CHAR(6)   NOT NULL
-- ,FOREIGN KEY (AKEY) REFERENCES FINOHDR ON DELETE CASCADE
  )
```

MERVA-Specific Tables

MERVA-specific tables can only be joined to the other S.W.I.F.T. Link tables by the UMR, because there is not necessarily a direct relationship to one transmitted message. If a message cannot be sent by SWIFT Link, for example because of a checking or authentication error, there will be one row in the MER table, but because the message has not yet been sent, it will not otherwise be recorded in the SWIFT Link tables. If the message is subsequently corrected and successfully sent, it will be recorded in the other SWIFT Link tables. The row in the MER table is related to the message, but not to the particular SWIFT Link event.

SRC Tables

The SRC tables (FINISRC, GPAISRC) list the MERVA ESA ready queues from which S.W.I.F.T. input APDU 01 messages are sent. Also, if a row is written to an MER table, a row is added to the corresponding SRC table.

Outgoing S.W.I.F.T. service messages are generated by SWIFT Link and therefore have no sending queue.

SRC tables have definitions of the form:

```
CREATE TABLE FINISRC
  ( PKEY            TIMESTAMP NOT NULL     -- insertion timestamp
   ,UMR             CHAR(28)  NOT NULL     -- MERVA Unique Msg.Ref.no.
   ,QUEUE           CHAR(8)   NOT NULL
   ,QSN             INTEGER   NOT NULL
  )
```

TAR Tables

The TAR tables (FINITAR, FINOTAR, GPAITAR, GPAOTAR) list the MERVA ESA target queues to which output messages and messages generated by SWIFT Link are routed after processing by SWIFT Link. Also, if a row is written to an MER table, a row is added to the corresponding TAR table. Up to twelve rows can be inserted for one event.

TAR tables have definitions of the form:

```
CREATE TABLE FINITAR
  ( PKEY            TIMESTAMP NOT NULL     -- insertion timestamp
   ,UMR             CHAR(28)  NOT NULL     -- MERVA Unique Msg.Ref.no.
   ,QUEUE           CHAR(8)   NOT NULL
   ,QSN             INTEGER   NOT NULL
  )
```

MER Tables

The MER tables (FINIMER, GPAIMER) contain MERVA ESA error messages for S.W.I.F.T. messages that could not be input to the S.W.I.F.T. network.

MER tables have definitions of the form:

```
CREATE TABLE FINIMER
  ( PKEY            TIMESTAMP NOT NULL     -- insertion timestamp
   ,UMR             CHAR(28)  NOT NULL     -- MERVA Unique Msg.Ref.no.
   ,ERROR           VARCHAR(   80) NOT NULL
  )
```

Queue Monitor Tables

The following figure shows the tables used by the queue monitor.

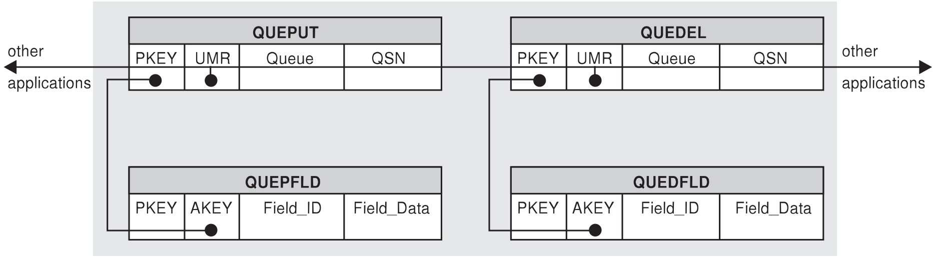


Figure 6. Queue Monitor Tables

For each queue event, a row is inserted into QUEDEL or QUEPUT. The QUEPUT and QUEDEL rows for a single message are related via their UMR columns.

Any fields can be extracted from a queue event (that is, from the queue element) and added to QUEDFLD or QUEPFLD. This requires that an MCB and a scanner control table be defined, and that they be named in the reconciliation parameter module IMRPRM. QUEDFLD is related to QUEDEL, and QUEPFLD to QUEPUT, via their AKEY columns.

QUEPUT Table

When a message is put into a MERVA ESA queue, a row is inserted into QUEPUT. The queue must be named in the Traffic Reconciliation parameters module IMRPRM with **SCOPE=BOTH** or **SCOPE=PUT**.

The definition of QUEPUT has the form:

```
CREATE TABLE QUEPUT
  ( PKEY        TIMESTAMP     NOT NULL     -- Timestamp of insertion
   ,EKEY        TIMESTAMP     NOT NULL     -- Timestamp of event
   ,QUEUE       CHAR(8)       NOT NULL     -- Queue name
   ,QSN         INTEGER       NOT NULL     -- Queue seq number
   ,UMR         CHAR(28)      NOT NULL     -- MERVA Unique Msg Ref.
   ,PRIMARY KEY (PKEY)
  )
```

QUEDEL Table

When a message is deleted from a MERVA ESA queue, a row is inserted into QUEDEL. The queue must be named in the Traffic Reconciliation parameters module IMRPRM with **SCOPE=BOTH** or **SCOPE=DELETE**.

The definition of QUEDEL has the same form as the definition of QUEPUT.

QUEPFLD Table

TOF fields from a message put into a MERVA ESA queue can be recorded in QUEPFLD, which is a dependent table of QUEPUT.

The definition of QUEPFLD has the form:

```
CREATE TABLE QUEPFLD
  ( PKEY        TIMESTAMP     NOT NULL     -- Timestamp of insertion
   ,AKEY        TIMESTAMP     NOT NULL     -- PKEY in QUEPUT
   ,FIELD_ID    CHAR(8)       NOT NULL     -- Field identification
   ,FIELD_DATA  VARCHAR(256)  NOT NULL     -- Field data
-- ,FOREIGN KEY (AKEY) REFERENCES QUEPUT ON DELETE CASCADE
  )
```

QUEDFLD Table

TOF fields from a message that is deleted from a MERVA ESA queue can be recorded in QUEDFLD, which is a dependent table of QUEDEL.

The definition of QUEDFLD has the same form as the definiton of QUEPFLD.

Journal Monitor Table

There is only one DB2 table for journal events.

JRNPUT Table

When a record is written to the MERVA ESA journal, if that record's identifier has been specified in the JRNID parameter of the Traffic Reconciliation parameters module IMRPRM, then a row is inserted into JRNPUT.

The definition of JRNPUT is:

```
CREATE TABLE JRNPUT
  ( PKEY       TIMESTAMP      NOT NULL     -- Timestamp insertion
   ,EKEY       TIMESTAMP      NOT NULL     -- Timestamp event
   ,JRN_ID     CHAR(1)        NOT NULL     -- Record identifier
   ,JRN_DATE   CHAR(8)        NOT NULL     -- Record date
  ,JRN_TIME   CHAR(6)        NOT NULL     -- Record time
  ,JRN_USER   CHAR(25)       NOT NULL     -- User extension
  ,JRN_DATA   VARCHAR(16000) NOT NULL     -- Record data
  ,PRIMARY KEY (PKEY)
 )
```

Note that the JRN_ID column contains the binary ID from the journal record. You would normally use the SQL HEX() function when retrieving it.

MERVA Link Monitor Tables

The following figure shows the structure of the output tables used by the MLINK monitor; the input tables have the same structure.

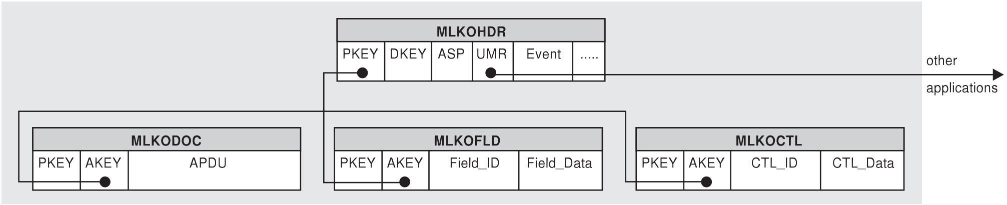


Figure 7. The Output MERVA Link Monitor Tables

In the MERVA Link tables, the HDR table is the root table. The DOC, FLD, and CTL tables are dependent tables related to the HDR table via their AKEY columns.

MLKxHDR Tables

The input and output HDR tables (MLKIHDR and MLKOHDR) contain MERVA ESA and MERVA Link identification data.

ASP_NAME Name of the MERVA Link ASP that is being monitored.

EVENT Category of the event (S, O, C, I, R, V):

S	Ready to send message
O	Outgoing message
C	Confirmed message
I	Incoming application message
R	Status report
V	Recovered or rerouted message

See the *MERVA for ESA V4 Customization Guide* for a detailed description of the possible events.

MSG_CATEGORY
Either a message (M) or a status report (A).

UMR MERVA ESA UMR of the message or report.

MSG_TYPE MERVA ESA message type.

MSG_STATUS
MERVA Link message processing status (the TOF field EKACLASS).

MSG_IAM MERVA Link message identification (the TOF field EKAAMSID).

These tables have definitions of the form:

```
CREATE TABLE MLKIHDR
  ( PKEY           TIMESTAMP NOT NULL
   ,EKEY           TIMESTAMP NOT NULL     -- Event time
   ,ASP_NAME       CHAR(8)   NOT NULL
   ,EVENT          CHAR(1)   NOT NULL     -- Event type S/O/C/I/R/V
   ,MSG_CATEGORY   CHAR(1)   NOT NULL     -- Message/ack (M/A)
   ,UMR            CHAR(28)  NOT NULL     -- Merva UMR
   ,MSG_TYPE       CHAR(8)   NOT NULL     -- Msgtype in DSLEXIT
   ,MSG_STATUS     CHAR(2)   NOT NULL     -- EKACLASS
   ,MSG_IAM        CHAR(16)  NOT NULL     -- IA message ID
   ,PRIMARY KEY (PKEY)
   )
```

MLKxDOC Tables

The input and output DOC tables (MLKIDOC and MLKODOC) record MERVA Link messages in the format defined by the DOC parameter in the parameter module IMRPRM.

The length of the APDU column must not be less than the **MAXLEN** value in the table filter entry for this table.

These tables have definitions of the form:

```
CREATE TABLE MLKIDOC
  ( PKEY           TIMESTAMP NOT NULL
   ,AKEY           TIMESTAMP NOT NULL     -- Key in header table
   ,APDU           VARCHAR(11000) NOT NULL--
-- ,FOREIGN KEY (AKEY) REFERENCES MLKIHDR ON DELETE CASCADE
   )
```

MLKxFLD Tables

The input and output FLD tables (MLKIFLD and MLKOFLD) record fields from the application messages transferred by MERVA Link. Which fields are recorded is determined by the MCB and scanner control table named in the FLD parameter in the parameter module IMRPRM, and can be further controlled by specifications in the table filter.

The length of the FIELD_DATA column must not be less than the **MAXLEN** value in the table filter entry for this table.

These tables have definitions of the form:

```
CREATE TABLE MLKIFLD
  ( PKEY           TIMESTAMP NOT NULL     -- Primary key
   ,AKEY           TIMESTAMP NOT NULL     -- Key in header table
   ,FIELD_ID       CHAR(8)   NOT NULL     -- Field identification
   ,FIELD_DATA VARCHAR(256) NOT NULL      -- Value of data area
-- ,FOREIGN KEY (AKEY) REFERENCES MLKIHDR ON DELETE CASCADE
   )
```

MLKxCTL Tables

The input and output CTL tables (MLKICTL and MLKOCTL) are intended for system fields from MERVA Link events. Which fields are recorded is determined by the MCB and scanner control table named in the CTL parameter in the parameter module IMRPRM, and can be further controlled by specifications in the table filter.

The length of the COL_DATA column must not be less than the **MAXLEN** value in the table filter entry for this table.

These tables have definitions of the form:

```
CREATE TABLE MLKICTL
  ( PKEY           TIMESTAMP NOT NULL     -- Primary key
   ,AKEY           TIMESTAMP NOT NULL     -- Key in header table
   ,CTL_ID         CHAR(8)   NOT NULL     -- Field identification
   ,CTL_DATA   VARCHAR(256) NOT NULL      -- Value of data area
-- ,FOREIGN KEY (AKEY) REFERENCES MLKIHDR ON DELETE CASCADE
   )
```

Telex Link Monitor Tables

The following figures show the tables used by the telex monitor:

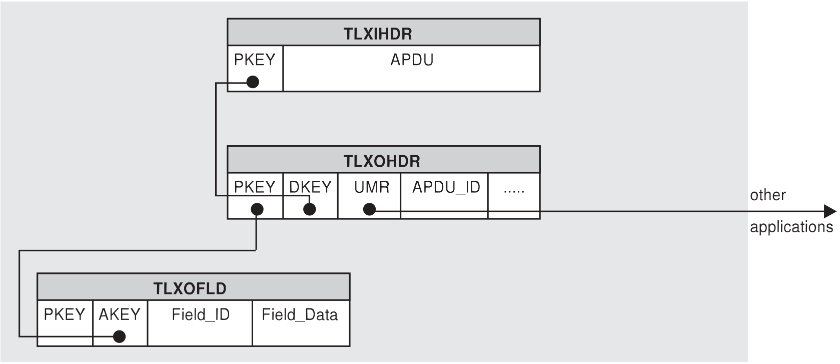


Figure 8. The Input Telex Link Monitor Tables

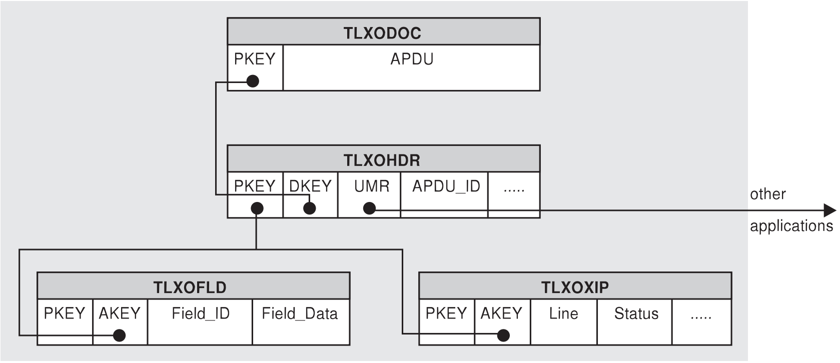


Figure 9. The Output Telex Link Monitor Tables

The DOC table is the root table. The HDR table is related to the DOC table by its DKEY column, and the FLD and TLXOXIP tables are related to the HDR table by their AKEY columns.

TLXxDOC Tables

The telex input and output documentation tables (TLXIDOC and TLXODOC) contain the APDUs transferred between MERVA ESA and Headoffice Telex on a fault-tolerant system:

- APDUs sent from MERVA ESA to Headoffice Telex on a fault-tolerant system are recorded in TLXIDOC.
- APDUs from Headoffice Telex on a fault-tolerant system to MERVA ESA are recorded in TLXODOC.

These APDUs are described in the *MERVA for ESA V4 Installation Guide*.

These tables have definitions of the form:

```
CREATE TABLE TLXIDOC
  ( PKEY            TIMESTAMP NOT NULL     -- Timestamp of insertion
   ,APDU       VARCHAR(22000) NOT NULL     -- DTNL-TXIP APDU
   ,PRIMARY KEY (PKEY)
   )
```

TLXxHDR Tables

The HDR tables (TLXIHDR, TLXOHDR) contain information from the telex APDUs. See the *MERVA for ESA V4 Installation Guide* for a description of the fields.

APDU_ID The identifier of the APDU. It is formed from the first character of the APDU and the 5-character identifier at character position 18 in the APDU.

EXCEPTION Contains 'Y' when an exception was detected:

- A logical acknowledgment indicates a duplicate telex.
- A received telex was truncated.

Otherwise it contains a blank.

The other columns are self-explanatory.

These tables have definitions of the form:

```
CREATE TABLE TLXIHDR
  ( PKEY            TIMESTAMP NOT NULL     -- Timestamp of insertion
   ,EKEY            TIMESTAMP NOT NULL     -- Event time
   ,DKEY            TIMESTAMP NOT NULL     -- Key in DOC
   ,UMR             CHAR(28)  NOT NULL     -- Merva UMR
   ,APDU_ID         CHAR(6)   NOT NULL     -- APDU identifier
   ,STATION         CHAR(8)   NOT NULL     -- Station name
   ,SESSION         CHAR(4)   NOT NULL     -- Session number
   ,SEQUENCE        CHAR(4)   NOT NULL     -- Sequence number
   ,O_SESSION       CHAR(4)   NOT NULL     -- Original session
   ,O_SEQUENCE      CHAR(4)   NOT NULL     -- Original sequence
   ,ACK_ID          CHAR(3)   NOT NULL     -- ACK/NAK indicator
   ,ACK_REASON      CHAR(4)   NOT NULL     -- ACK/NAK reason
   ,EXCEPTION       CHAR(1)   NOT NULL     -- Exception indicator
   ,PRIMARY KEY (PKEY)
-- ,FOREIGN KEY (DKEY) REFERENCES TLXIDOC ON DELETE CASCADE
   )
```

TLXxFLD Tables

Any fields in the message can be extracted and inserted into the FLD tables (TLXIFLD, TLXOFLD). This requires that an MCB and a scanner control table be defined, and that they be named in the reconciliation parameter module IMRPRM.

These tables have definitions of the form:

```
CREATE TABLE TLXIFLD
  ( PKEY            TIMESTAMP NOT NULL     -- Timestamp of insertion
   ,AKEY            TIMESTAMP NOT NULL     -- Key in HDR
   ,FIELD_ID        CHAR(3)   NOT NULL     -- Field identification
   ,FIELD_DATA  VARCHAR(256)  NOT NULL     -- Field data
-- ,FOREIGN KEY (AKEY) REFERENCES TLXIHDR ON DELETE CASCADE
   )
```

TLXOXIP Table

The table TLXOXIP lists status information for Telex Link via a fault-tolerant system from the following APDUs:

@SS001 Signon ACK/NAK

@SS002	Signoff ACK
’MS001	Logical ACK
!RR002	Transmission ACK/NAK
$SM001	Received telex
!RR999	Status report

The fields are described in the *MERVA for ESA V4 Installation Guide*.

The definition of TLXOXIP is:

```
CREATE TABLE TLXOXIP
  ( PKEY           TIMESTAMP NOT NULL     -- Timestamp of insertion
   ,AKEY           TIMESTAMP NOT NULL     -- Key in HDR
   ,TXNUMR         CHAR(20)  NOT NULL     -- Network response
   ,TXISEQ         CHAR(5)   NOT NULL     -- Document number
   ,TXCOUNT        CHAR(5)   NOT NULL     -- Telex count
   ,TXATIME        CHAR(12)  NOT NULL     -- Time last access
   ,TXETIME        CHAR(12)  NOT NULL     -- Time transmission end
   ,TXSTATUS       CHAR(2)   NOT NULL     -- Status telex
   ,TXSLINE        CHAR(1)   NOT NULL     -- Selected line number
   ,TXDURM         CHAR(5)   NOT NULL     -- Duration minutes
   ,TXDURS         CHAR(2)   NOT NULL     -- Duration seconds
   ,TRLINE         CHAR(2)   NOT NULL     -- Line number received
   ,TRTIME         CHAR(12)  NOT NULL     -- Telex time stamp
   ,TXIP_STATUS    CHAR(5)   NOT NULL     -- TXIP status
-- ,FOREIGN KEY (AKEY) REFERENCES TLXOHDR ON DELETE CASCADE
   )
```

MERVA-MQI Attachment Monitor Tables

The design of the MERVA-MQI Attachment monitor tables is similar to that of other monitors in that data is stored in DOC, HDR, and FLD tables. However, because of the way messages are transferred by MQSeries, the DOC table is in two parts:

- A DOC table for the customary message string
- A documentation field (DOF) table for messages with fields attached to the message string

The structure of MQI messages is described in the *MERVA for ESA V4 Customization Guide*.

The MQI HDR table is the root table. Other tables are related to it via their AKEY columns.

The following figure shows the input MERVA-MQI Attachment monitor tables. The output tables have a similar structure.

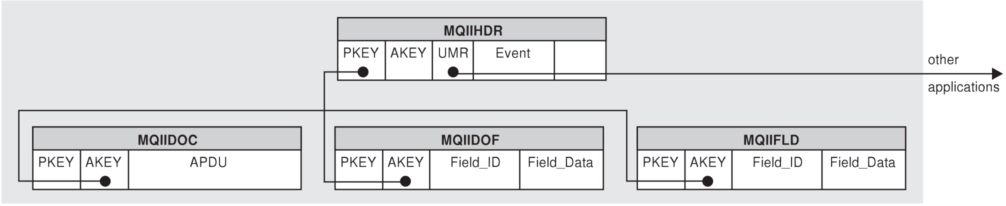


Figure 10. The Input MERVA-MQI Attachment Monitor Tables

MQIxHDR Tables

The HDR tables (MQIIHDR, MQIOHDR) contain columns identifying the event (UMR, process name, event type) and status fields from the MQI MQMD and MQPMO or MQGMO structures.

These tables have definitions of the form:

```
CREATE TABLE MQIIHDR
  ( PKEY          TIMESTAMP NOT NULL   --Primary key, insertion time
   ,EKEY          TIMESTAMP NOT NULL   --Event time
   ,MTYPE         CHAR(8)   NOT NULL   --MERVA Msg.Type
   ,UMR           CHAR(28)  NOT NULL   --MERVA Unique Msg.Ref
   ,PROCESS       CHAR(8)   NOT NULL   --Process name
   ,EVENT         CHAR(8)   NOT NULL   --MQ Msgtype (DATAGRAM,REQUEST..)
   ,DOC_FIELDS    INTEGER   NOT NULL   --No.of DOF table rows
--   MQMD & MQPMO structure fields:
   ,REPORT        INTEGER   NOT NULL   --Report options
   ,EXPIRY        INTEGER   NOT NULL   --Expiry time
   ,FEEDBACK      INTEGER   NOT NULL   --Report feedback
   ,ENCODING      INTEGER   NOT NULL   --Data encoding
   ,CCSID         INTEGER   NOT NULL   --Coded character set
   ,FORMAT        CHAR(8)   NOT NULL   --(MQ) format name
   ,PRIORITY      INTEGER   NOT NULL   --Priority
   ,PERSISTENCE   INTEGER   NOT NULL   --Persistence
   ,MSGID         CHAR(24)  NOT NULL   --(MQ) Message identifier
   ,CORRELID      CHAR(24)  NOT NULL   --Correlation identifier
   ,BACKOUT       INTEGER   NOT NULL   --MQGET Backout count
   ,USERID        CHAR(12)  NOT NULL   --Originating user
   ,ACCOUNTING    CHAR(32)  NOT NULL   --Appl.Accounting token
   ,APPLIDENT     CHAR(32)  NOT NULL   --Application identity data
   ,PUTAPPLTYPE   INTEGER   NOT NULL   --Originating appl.type
   ,PUTAPPLNAME   CHAR(28)  NOT NULL   --Originating appl.name
   ,PUTDATE       CHAR(8)   NOT NULL   --Originating date
   ,PUTTIME       CHAR(8)   NOT NULL   --Originating time
   ,APPLORIGIN    CHAR(4)   NOT NULL   --Originating appl.data
   ,OPTIONSID     CHAR(4)   NOT NULL   --'PMO ' or 'GMO '
   ,OPTIONS       INTEGER   NOT NULL   --(MQPMO/MQGMO) Msg.options
   ,WAITINTERVAL  INTEGER   NOT NULL   --(MQGMO) MQGET wait interval
   ,REPLYTOQ      VARCHAR(48) NOT NULL --Reply-to queue
   ,REPLYTOQMGR   VARCHAR(48) NOT NULL --Reply-to queue manager
   ,RESOLVEDQ     VARCHAR(48) NOT NULL --(MQPMO/MQGMO) Dest.queue
   ,RESOLVEDQMGR  VARCHAR(48) NOT NULL --(MQPMO) Dest.queue mgr
   ,PRIMARY KEY (PKEY)
   )
```

An MQIxHDR_TEMP table is also defined. This is used internally by the MQI table insertion process when inserting events asynchronously from the flip-flop data set. It enables Traffic Reconciliation to ensure that the MQIIHDR table only contains events committed by MQSeries. After an MQSeries commit event has been processed the MQIxHDR_TEMP table will be empty.

The MQIxHDR_TEMP table is not used when events are inserted synchronously to DB2 by the MQI extraction exit.

MQIxDOC Tables

DOC tables contain the application message from a datagram, request, or report message. A row is also written for a reply message but the APDU column will contain a string of length 0; a reply consists only of one or more fields, all of which are stored in the DOF table.

These tables have definitions of the form:

```
CREATE TABLE MQIIDOC
  ( PKEY          TIMESTAMP NOT NULL      --Primary key, insertion time
   ,AKEY          TIMESTAMP NOT NULL      --HDR key
   ,APDU          VARCHAR(11000) NOT NULL --Appl.string
-- ,FOREIGN KEY (AKEY) REFERENCES MQIIHDR ON DELETE CASCADE
  )
```

MQIxDOF Tables

The DOF tables (MQIIDOF, MQIODOF) contain any fields attached to the application message. One row is inserted for each field.

These tables have definitions of the form:

```
CREATE TABLE MQIIDOF
  ( PKEY          TIMESTAMP NOT NULL     --Primary key, insertion time
   ,AKEY          TIMESTAMP NOT NULL     --HDR key
   ,FIELD_ID      CHAR(8)   NOT NULL     --Field ID
   ,FIELD_DATA    VARCHAR(256) NOT NULL  --Field data
-- ,FOREIGN KEY (AKEY) REFERENCES MQIIHDR ON DELETE CASCADE
  )
```

MQIxFLD Tables

Any fields from the event TOF can be extracted to the FLD input and output tables (MQIIFLD and MQIOFLD). Which fields are recorded is determined by the MCB and scanner control table named in the FLD parameter in the parameter module IMRPRM, and can be further controlled by specifications in the table filter.

These tables have definitions of the form:

```
CREATE TABLE MQIIFLD
  ( PKEY          TIMESTAMP NOT NULL     --Primary key, insertion time
   ,AKEY          TIMESTAMP NOT NULL     --HDR key
   ,FIELD_ID      CHAR(8)   NOT NULL     --Field ID
   ,FIELD_DATA    VARCHAR(256) NOT NULL  --Field data
-- ,FOREIGN KEY (AKEY) REFERENCES MQIIHDR ON DELETE CASCADE
  )
```

Other DB2 Tables

IMRCTL Table

The IMRCTL table is used internally by Traffic Reconciliation to support restart of the insertion of events from the flip-flop data set. Columns LSTRBA and LSTFID identify the most recent event record committed by DB2.

LSTMRV The **NAME=** parameter from the MERVA ESA parameters module DSLPRM accessed by the insertion transaction. Traffic Reconciliation supports capture of events from multiple MERVA ESA systems, using separate flip-flop files and separate insertion transactions, into a common DB2 database.

There is one row in IMRCTL for each supported MERVA system.

LSTRBA The RBA of the last flip-flop event record inserted into the DB2 tables and committed by DB2.

LSTFID The identifier of the flip-flop data set containing the last committed event record. This identifier is a timestamp that indicates the time when the first event in the flip-flop data set was written, and can be found in the first record of the data set, and in the control record in the flip-flop IMRCTL data set.

Part 2. Using Traffic Reconciliation

Chapter 3. Issuing Queries

Queries can be of two types:

- MCB-based queries
- QMF queries

Issuing MCB-Based Queries

An MCB-based query is a special kind of MERVA message that is used to create an SQL query. The resulting SQL queries are processed asynchronously by the batch SQL processor program (IMRSQLP), and the results are returned as MERVA messages to a MERVA queue. The results of an MCB-based query can be one of the following:

- One or more messages, retrieved from the database, that meet the selection criteria. These messages can be displayed using the messages' MCBs.
- A single message containing an SQL result table. Such a query is called a *list query*. Displaying the results of a list query requires a special MCB that must be provided by your installation.

Because MCB-based queries use the MERVA ESA MCB mechanism, they can be integrated into the MERVA ESA end user interface:

- Each query can provide a data entry panel for setting query parameters.
- Default and checking exits can be used to set and validate query parameters.

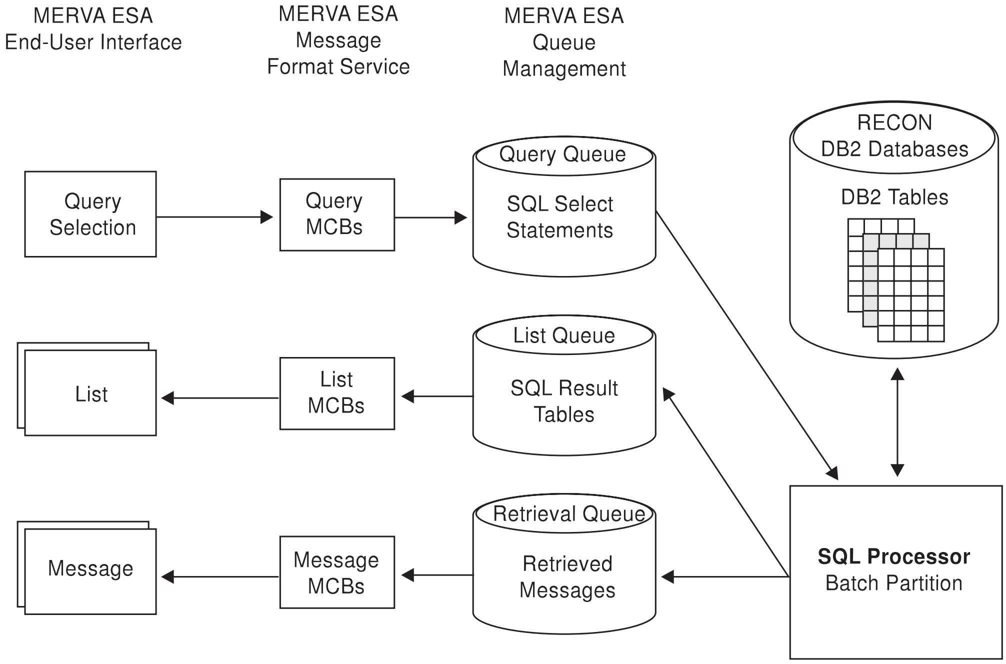


Figure 11. MCB-Based Queries

MERVA ESA provides the following sample MCB-based queries:

Q001 List FIN input/output messages

Q002 List journal records

Q003 List FIN input MT 100 amounts

Q004 List MLINK traffic for specific ASPs

Q005 S.W.I.F.T. input messages + ACK/NAK

Q006 List of DB2 tables attributes

Q007 Retrieve S.W.I.F.T. messages for specified correspondent and amount

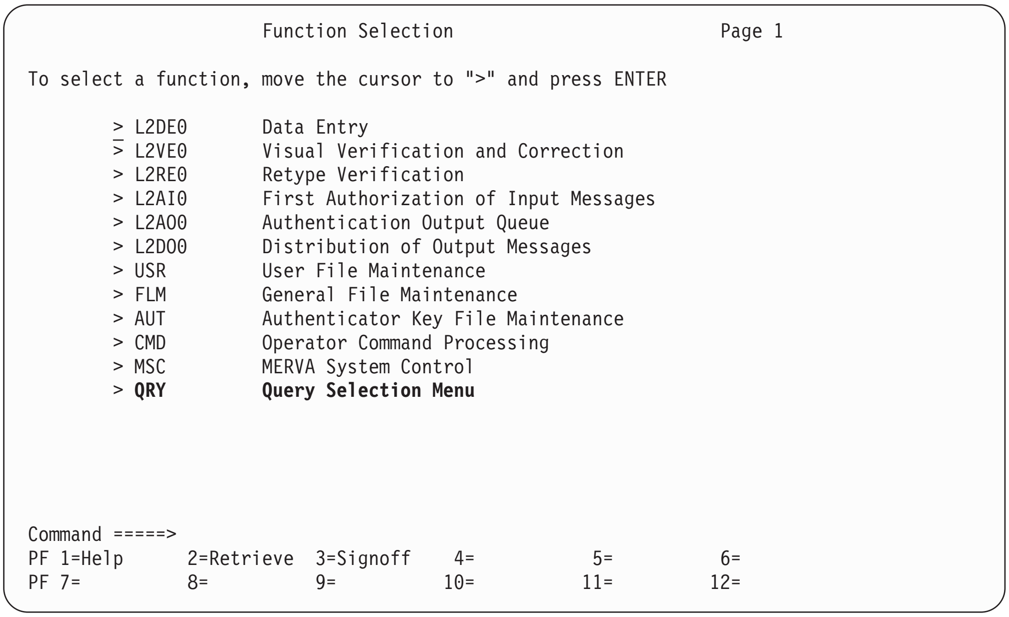
Q008 Retrieve S.W.I.F.T. messages for specified ISN

Q009 Retrieve S.W.I.F.T. messages for specified OSN

Q010 Retrieve S.W.I.F.T. messages for specified TRN

Initiating Queries

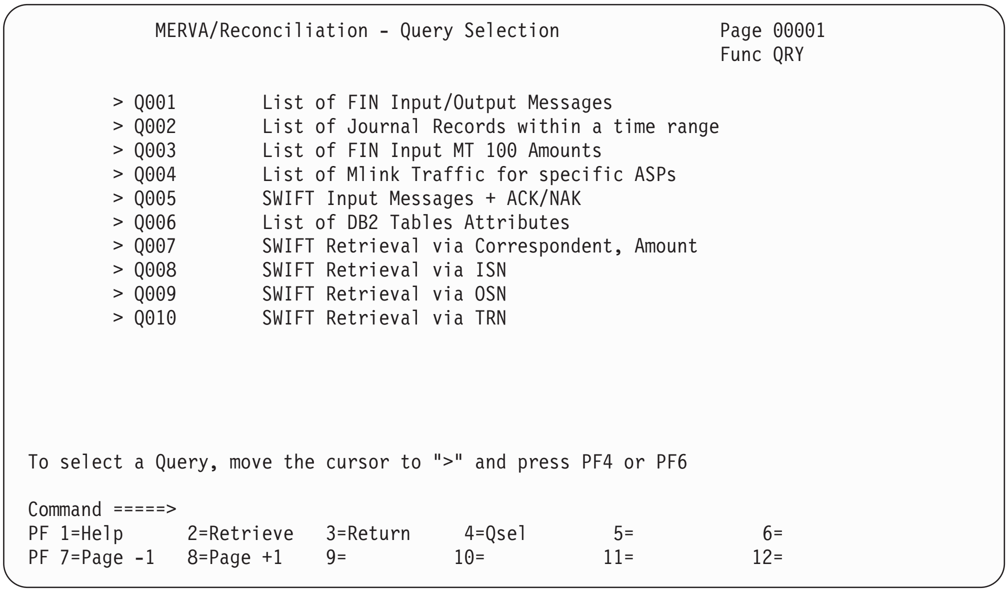
To initiate an MCB-based query, select QRY from the main MERVA function-selection panel, an example of which is shown in Figure 12.



```
                        Function Selection                         Page 1

 To select a function, move the cursor to ">" and press ENTER

        > L2DE0       Data Entry
        > L2VE0       Visual Verification and Correction
        > L2RE0       Retype Verification
        > L2AI0       First Authorization of Input Messages
        > L2AO0       Authentication Output Queue
        > L2DO0       Distribution of Output Messages
        > USR         User File Maintenance
        > FLM         General File Maintenance
        > AUT         Authenticator Key File Maintenance
        > CMD         Operator Command Processing
        > MSC         MERVA System Control
        > QRY         Query Selection Menu





 Command =====>
 PF 1=Help      2=Retrieve  3=Signoff    4=          5=          6=
 PF 7=          8=          9=          10=         11=         12=
```

Figure 12. MERVA Function Selection Menu

MERVA displays the query selection menu (see Figure 13 on page 35), which lists all the queries you are authorized to run.



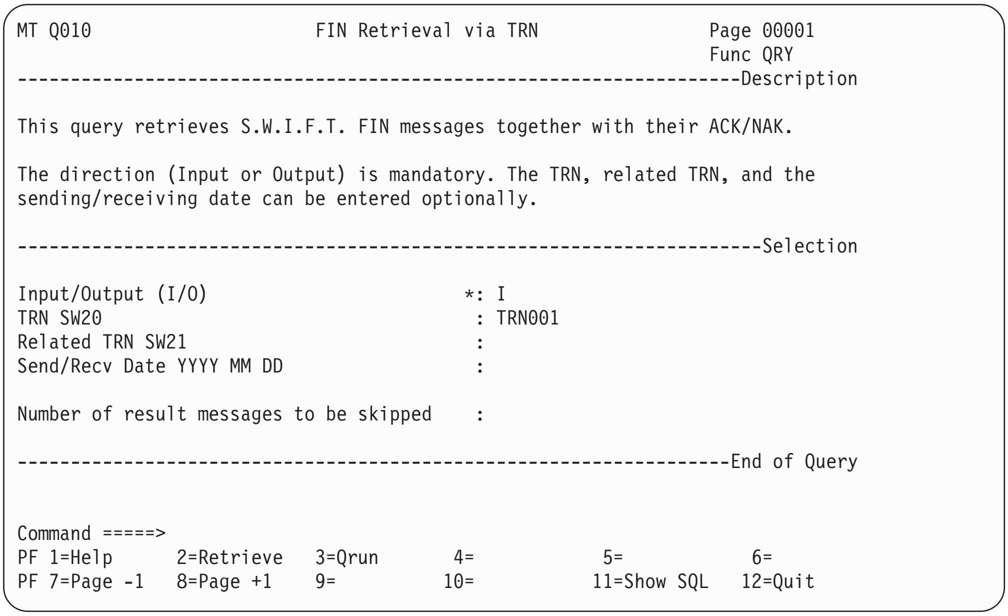
```
              MERVA/Reconciliation - Query Selection              Page 00001
                                                                  Func QRY

          > Q001        List of FIN Input/Output Messages
          > Q002        List of Journal Records within a time range
          > Q003        List of FIN Input MT 100 Amounts
          > Q004        List of Mlink Traffic for specific ASPs
          > Q005        SWIFT Input Messages + ACK/NAK
          > Q006        List of DB2 Tables Attributes
          > Q007        SWIFT Retrieval via Correspondent, Amount
          > Q008        SWIFT Retrieval via ISN
          > Q009        SWIFT Retrieval via OSN
          > Q010        SWIFT Retrieval via TRN





 To select a Query, move the cursor to ">" and press PF4 or PF6

 Command =====>
 PF 1=Help      2=Retrieve   3=Return     4=Qsel        5=           6=
 PF 7=Page -1   8=Page +1    9=          10=           11=          12=
```

Figure 13. Query Selection Menu

Select a query by doing one of the following:

- Place the cursor on the appropriate line and press PF4.
- Enter **QSEL xxxx** on the command line, where **xxxx** is the query identifier (for example, Q001).

MERVA displays the query input panel. For example, the input panel for Q010 is shown in Figure 14.



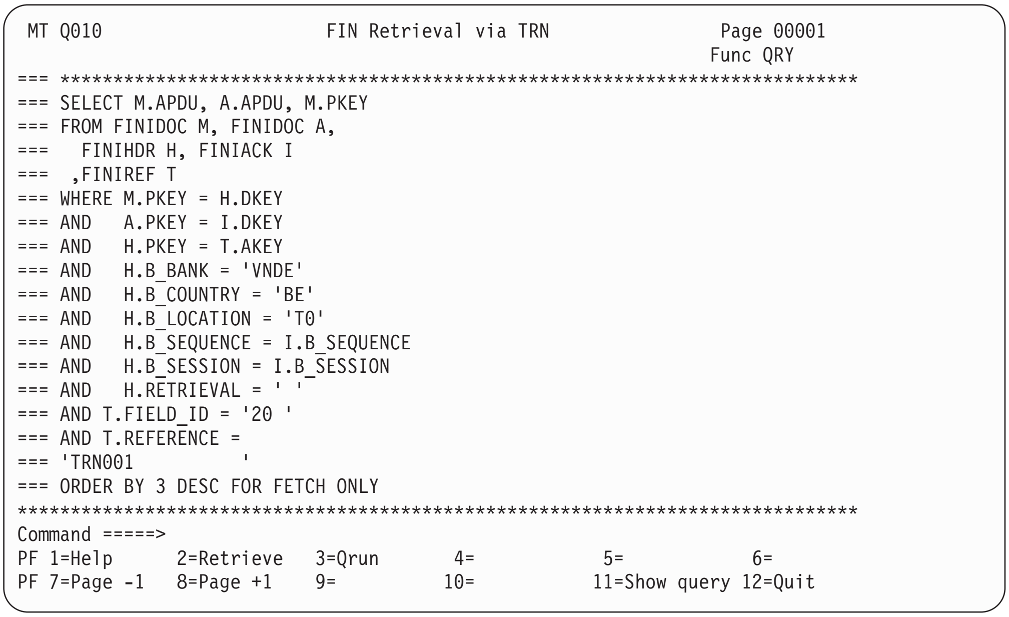
```
MT Q010                     FIN Retrieval via TRN              Page 00001
                                                               Func QRY
-------------------------------------------------------------------Description

This query retrieves S.W.I.F.T. FIN messages together with their ACK/NAK.

The direction (Input or Output) is mandatory. The TRN, related TRN, and the
sending/receiving date can be entered optionally.

---------------------------------------------------------------------Selection

Input/Output (I/O)                        *: I
TRN SW20                                   : TRN001
Related TRN SW21                           :
Send/Recv Date YYYY MM DD                  :

Number of result messages to be skipped    :

------------------------------------------------------------------End of Query


Command =====>
PF 1=Help      2=Retrieve   3=Qrun       4=           5=           6=
PF 7=Page -1   8=Page +1    9=          10=          11=Show SQL   12=Quit
```

Figure 14. Query Input Panel

Enter the parameters for the query.

To view the SQL SELECT statement that will be generated by the query, press PF11, or enter **NOPROMPT** on the command line. MERVA displays a panel similar to the one shown in Figure 15. To return to the parameter input panel press PF11 again, or enter the **PROMPT** command.

To submit the query, press PF3, or enter **QRUN** on the command line. The query is submitted to the SQL processor program (IMRSQLP) for processing, and the results are returned in MERVA queues as determined by the query table (IMRQRYT).



```
 MT Q010                     FIN Retrieval via TRN                Page 00001
                                                                 Func QRY
=== ***************************************************************************
=== SELECT M.APDU, A.APDU, M.PKEY
=== FROM FINIDOC M, FINIDOC A,
===   FINIHDR H, FINIACK I
===  ,FINIREF T
=== WHERE M.PKEY = H.DKEY
=== AND   A.PKEY = I.DKEY
=== AND   H.PKEY = T.AKEY
=== AND   H.B_BANK = 'VNDE'
=== AND   H.B_COUNTRY = 'BE'
=== AND   H.B_LOCATION = 'T0'
=== AND   H.B_SEQUENCE = I.B_SEQUENCE
=== AND   H.B_SESSION = I.B_SESSION
=== AND   H.RETRIEVAL = ' '
=== AND T.FIELD_ID = '20 '
=== AND T.REFERENCE =
=== 'TRN001          '
=== ORDER BY 3 DESC FOR FETCH ONLY
*******************************************************************************
Command =====>
PF 1=Help      2=Retrieve   3=Qrun       4=            5=            6=
PF 7=Page -1   8=Page +1    9=          10=           11=Show query 12=Quit
```

Figure 15. Displaying the Generated SQL Statement

PF Key Commands

These commands are assigned to PF keys.

QSEL Command: This command can be issued within the query selection function or during the display of a query.

qsel **qs**	{*queryid*}

Use this command to select a query. The parameter specifies the query's message identifier. If the command is issued during the display of a query, the current query is lost and the specified query is displayed.

QRUN Command: This command can be issued within the query selection function or during the display of a query.

qrun **qr**	[*queryid*]

Use this command to run a query from the query selection menu or to run the current query displayed on the screen. To run a query from the selection menu, specify the query identifier (message identifier of the query's MCB). Otherwise the current query is run.

QQUIT Command: This command can be issued during the display of a query.

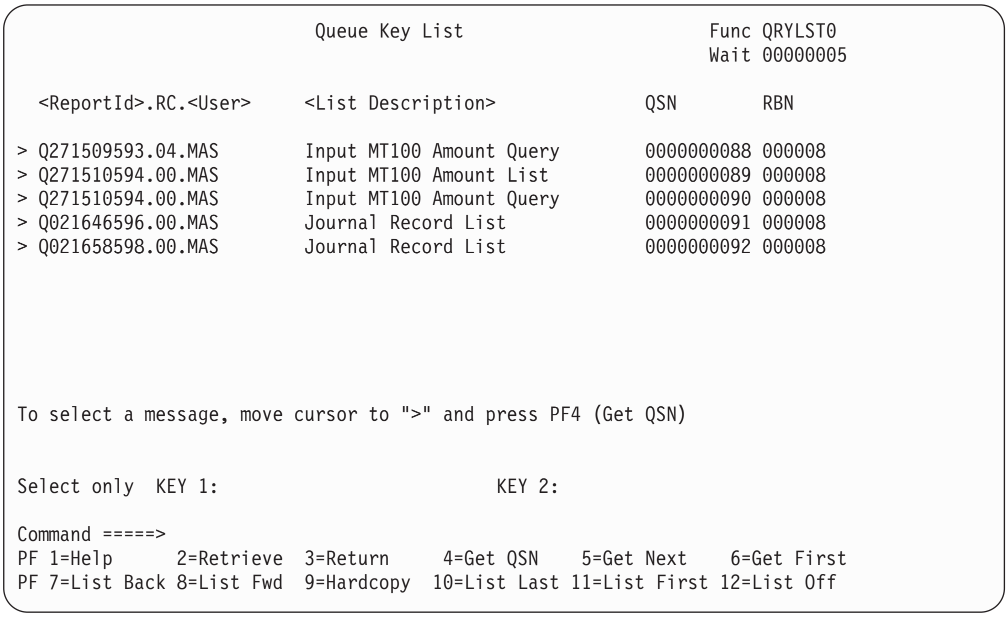
qquit **qq**	

Use this command to quit the current query displayed on the screen and return to the query selection panel.

Viewing Results of Queries

The SQL processor program (IMRSQLP) operates in the background processing queries. Depending on the nature of a query, IMRSQLP retrieves the selected messages, or generates a new message containing a result table. These messages are written to MERVA queues according to the message routing defined in the query table. In addition, for each query IMRSQLP processes, it writes a query-response message to a MERVA queue. A query-response message contains a copy of the original query, plus return code information regarding the processing.

All of these messages (retrieved messages, messages containing result tables, query-response messages) are MERVA messages and can be processed further using common MERVA functions. Erroneous and rejected queries are placed into the SQL processor's error queue. Figure 16 shows the result of a queue LIST command for a queue containing the results of list queries.



```
                            Queue Key List                       Func QRYLST0
                                                                 Wait 00000005

  <ReportId>.RC.<User>     <List Description>             QSN        RBN

> Q271509593.04.MAS        Input MT100 Amount Query       0000000088 000008
> Q271510594.00.MAS        Input MT100 Amount List        0000000089 000008
> Q271510594.00.MAS        Input MT100 Amount Query       0000000090 000008
> Q021646596.00.MAS        Journal Record List            0000000091 000008
> Q021658598.00.MAS        Journal Record List            0000000092 000008







To select a message, move cursor to ">" and press PF4 (Get QSN)


Select only  KEY 1:                          KEY 2:

Command =====>
PF 1=Help      2=Retrieve  3=Return     4=Get QSN    5=Get Next    6=Get First
PF 7=List Back 8=List Fwd  9=Hardcopy  10=List Last 11=List First 12=List Off
```

Figure 16. List of Query Results

Deleting Results

The CLEAR Command

This command can be issued when accessing a message-processing function.

clear **clr**	

Use this command to delete all messages produced for you by the SQL processor. All lists, retrievals, and responses produced for queries from you are deleted from the message-processing function's queue.

The DELALL Command

This command can be issued when accessing a message-processing function.

<table>
<tr><td><b>delall</b>
<b>dall</b></td><td>{<i>key1value</i>} | {<i>,key2value</i>}</td></tr>
</table>

The command deletes from the queue all messages with a first key matching *key1value*, or a second key matching *key2value* (note the comma before *key2value*). The key value can contain the wildcard characters:

*	To represent any character string
%	To represent any single character

Issuing QMF Queries

MERVA provides about 50 predefined Query Management Facility (QMF) queries, which are described in "Appendix B. QMF Queries" on page 113. After installing them in QMF under TSO, you can invoke them from QMF in a TSO session. Alternatively, you can run them in a batch environment to produce printed reports. These queries support all monitors except MQI (no queries are supplied for MQI).

Issuing QMF Queries from TSO

For information on how to use QMF, refer to *Query Management Facility User Guide*.

Issuing QMF Queries by Running Batch Programs

For information on how to run QMF queries as batch programs to produce printed reports, refer to *Query Management Facility Managing QMF for MVS*.

QMF provides sample JCL in the data set **hlq.DSQSAMPE(DSQ1EINV)**. For example, to run query IMRQ05A using this JCL, you would need to run the corresponding QMF procedure (IMRP05A) by adding statements such as these to the SYSTSIN data set:

```
//SYSTSIN  DD  *
  ISPSTART PGM(DSQQMFE) NEWAPPL(DSQE) +
   PARM(M=B,DSQSRUN=IMRP05A(&&LOWUMRSEQ='0',+
                            &&HIGHUMRSEQ='1000'+
                            ),+
        S=DB2s
/*
```

Chapter 4. Imbedding S.W.I.F.T. Messages into Common Group Messages

You can imbed S.W.I.F.T. messages into S.W.I.F.T. common group messages of the type n92, n95, and n96. You can think of this as putting the S.W.I.F.T. message into an envelope to be stored or sent elsewhere.

In an end user driver terminal session, the imbedded message is protected from modification.

Imbedding Messages from the Database

To retrieve S.W.I.F.T. messages from the Traffic Reconciliation tables by transaction or sequence number, and to imbed them in n9x envelopes, issue the commands:

G92 To retrieve and imbed a message into an n92 message

G95 To retrieve and imbed a message into an n95 message

G96 To retrieve and imbed a message into an n96 message

You can issue these commands only from the message selection panel of a MERVA data entry function, and you can retrieve only messages that are associated with your origin identification as specified in your MERVA user profile.

The command parameters must uniquely identify the message to be retrieved. Traffic Reconciliation distinguishes a TRN from an OSN or ISN by the following rule: if it contains more than six characters or any nonnumeric character, it is a TRN; if it contains six or fewer characters, all numeric, it is an OSN or ISN.

After the command is issued, the message is retrieved from the DB2 database, enclosed in the appropriate message envelope, and written to the MERVA queue of the function from which the command was invoked. Traffic Reconciliation places the appropriate **GET QSN** command in the command line, so to view the constructed message all you need to do is to press the Enter key.

G9x Command Syntax

G92 Command

Use the following command to retrieve a S.W.I.F.T. input message by ISN and imbed it into an n92.

G92	**I**,*isn*

Use the following command to retrieve a S.W.I.F.T. input message by TRN and imbed it into an n92. If the message is not uniquely identified by the TRN, specify also the correspondent, with or without the branch code.

G92	**I**,*trn*[,*correspondent*[*branchcode*]]

G95 Command

Use the following command to retrieve a S.W.I.F.T. input message by ISN, a S.W.I.F.T. output message by OSN, or a S.W.I.F.T. output message by the sender's

ISN, and imbed it into an n95.

G95	**I**,*isn*
G95	**O**,*osn*
G95	**O**,*isn*,*corr*

Use the following command to retrieve a S.W.I.F.T. input or output message by TRN and imbed it into an n95. If the message is not uniquely identified by the TRN, specify also the correspondent, with or without the branch code.

G95	**I**,*trn*[,*correspondent*[*branchcode*]]
G95	**O**,*trn*[,*correspondent*[*branchcode*]]

G96 Command

Use the following command to retrieve a S.W.I.F.T. input message by ISN, a S.W.I.F.T. output message by OSN, or a S.W.I.F.T. output message by the sender's ISN, and imbed it into an n96.

G96	**I**,*isn*
G96	**O**,*osn*
G96	**O**,*isn*,*corr*

Use the following command to retrieve a S.W.I.F.T. input or output message by TRN and imbed it into an n96. If the message is not uniquely identified by the TRN, specify also the correspondent, with or without the branch code.

G96	**I**,*trn*[,*correspondent*[*branchcode*]]
G96	**O**,*trn*[,*correspondent*[*branchcode*]]

G9x Command Examples

Here are some examples of G9x commands:

```
G92 I,T001                  (1)
G95 O,1234                  (2)
G96 I,4002,CORRDEST         (3)
G96 O,5678,CORRDESTBBB      (4)
```

1. The S.W.I.F.T. input message with the TRN **T001** is retrieved and imbedded into an n92 envelope. The message is uniquely identified by the TRN.
2. The S.W.I.F.T. output message with the OSN **001234** is retrieved and imbedded into an n95 envelope. The OSN must uniquely identify the message. Because the character string following the **O** contains 6 or fewer characters, all numeric, it is recognized as not being a TRN.
3. The S.W.I.F.T. input message with the TRN **4002** that was sent to the correspondent destination **CORRDEST** is retrieved and imbedded into an n96 envelope. The TRN alone was not enough to uniquely identify the message, so the correspondent is specified as well.
4. The S.W.I.F.T. output message with the ISN **5678** of correspondent destination **CORRDEST** and branch code **BBB** is retrieved and imbedded into an n96 envelope.

Imbedding Messages from the Display

You can also imbed into an n9x message a S.W.I.F.T. input message that is currently being displayed; for example, one that was retrieved using an MCB-based query. To do this, enter one of the following commands on the command line of the display panel:

I92	[**X**]
I95	[**X**]
I96	[**X**]

Notes:

1. The headers of the common group message and the fields 21 and 11 are built based on the current message.
2. The date subfield of field 11S is extracted from field 177 of the message ACK contained in the TOF field MSGACK.
3. If you do not specify a parameter for this command, the text block fields of the current message are copied. If you specify the parameter **X**, no text block fields are copied; instead, an empty field 79 is inserted.

After the message is enclosed in the appropriate message envelope, it remains displayed and awaits further processing (for example, storage or sending).

Chapter 5. Operating Traffic Reconciliation

Starting and Stopping Reconciliation

After Traffic Reconciliation has been successfully installed, it becomes an integral part of MERVA ESA and cannot be stopped or restarted independently. The extraction of reconciliation events can only by inhibited by setting **RECON=NO** in the MERVA ESA parameters module (DSLPRM), and then regenerating DSLPRM.

Inserting Events from the Flip-Flop into DB2 Tables

Events in the flip-flop data sets are inserted into the DB2 tables by the event insertion transaction IMRI (program IMRINSP). There are two versions of this program:

- One that runs as a CICS® or IMS transaction
- One that runs as an MVS™ batch program

Batch Insertion Program

The version of IMRINSP that runs as an MVS batch program is intended for use when MERVA ESA is inactive; for example, during the nightly batch window. It terminates with an error message if run while MERVA ESA is active.

The following JCL is an example of how to invoke the batch program:

```
//INSERT EXEC PGM=IMRINSP,PARM='COMMIT=nnnn'
//STEPLIB  DD DSN=recon.SIMRLODB,DISP=SHR
//         DD DSN=merva.SDSLLODB,DISP=SHR
//         DD DSN=DB2.LOADLIB,DISP=SHR
//IMRCTL   DD DSN=recon.RECCTL,DISP=SHR  Flip-flop control data set
//IMRDSA   DD DSN=recon.RECDSA,DISP=SHR  Flip-flop file A
//IMRDSB   DD DSN=recon.RECDSB,DISP=SHR  Flip-flop file B
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//
```

The **COMMIT** parameter is a number from 1 to 9999 that specifies the number of flip-flop records to be processed before a DB2 commit is performed. If you do not specify the **COMMIT** parameter here, the value set for the **COMMIT** parameter in the module IMRPRM is used.

This program can return the following codes to the operating system:

0 The program terminated normally.

8 The program terminated following an unexpected error. An operator message describing the error was issued.

Insertion Transaction

The MERVA nucleus program IMRICON is provided to control initiation of the event insertion transaction IMRI (program IMRINSP). As a nucleus program, IMRICON can be started from the MERVA ESA end user CMD panel using the MERVA **START** command: **s imricon**, and stopped with the MERVA **STOP** command: **p imricon**. Use the **DP** (display program) command to view the status of nucleus programs.

IMRICON can be started automatically at MERVA startup by specifying **AUTO=YES** in the nucleus program table (DSLNPTT) entry for IMRICON.

When IMRICON is active, it initiates transaction IMRI periodically, as determined by the **TIV** parameter in the reconciliation parameters module IMRPRM.

After initiating a transaction IMRICON will not reinitiate it until the transaction has successfully completed. If the transaction fails, it will not be reinitiated by IMRICON. In this case, to get IMRICON to start initiating the transaction again, you must stop and restart IMRICON.

IMRICON uses the standard MERVA ESA mechanism for initiating a transaction: a transaction is defined for the MERVA function specified by the DSLFNT **TRAN** parameter, and this is the function that is started.

If IMRICON is inactive, you can use the MERVA **SF** (start function) command to start the function manually. The name of the function is specified by the IMRPRM **DUMQUE** parameter. The default name is IMRI:

```
sf imri
```

If you are using only synchronous insertion, that is, if no events are being written to the flip-flop, there is no need to start IMRINSP or IMRICON.

Operating the SQL Processor

The purpose of the SQL processor (IMRSQLP) is to assemble, interpret, and run MCB-based SQL queries, and to process DELALL and CLEAR requests.

The SQL processor is a long-running batch program that polls MERVA ESA for work according to polling frequencies specified in parameter SRVWAIT of the parameters module IMRPRM. When DB2 or MERVA is not available, it polls according to the second SRVWAIT polling frequency.

The following commands are provided to control the SQL processor:

CONN Connects the SQL processor to MERVA

DISC Disconnects the SQL processor from MERVA

TERM/STOP Stops the SQL processor

These commands must be entered on the MVS system console as a reply to the outstanding WTOR operator message issued by the program. If the SQL processor is connected to MERVA, the message identification of the WTOR is IMR700A, otherwise it is IMR701A.

Starting the SQL Processor

The SQL processor program is started as an MVS batch job. It is a long-running task with the following JCL:

```
//RSQL      JOB
//RSQLP  EXEC PGM=IMRSQLP,PARM='  '
//STEPLIB  DD DSN=recon.SIMRLODB,DISP=SHR    Reconciliation Loadlib
//         DD DSN=merva.SDSLLODB,DISP=SHR    MERVA ESA Loadlib
//         DD DSN=DB2.LOADLIB,DISP=SHR       DB2 Loadlib
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//
```

The program can return the following codes to the operating system:

0 The program was terminated by the system operator, or because the IMRPRM **SRVSTOP** count was exhausted.

8 The program terminated following an unexpected error. An operator message describing the error has been issued.

Tracing the SQL Processor

MERVA activity in the SQL processor can be traced by activating the MERVA MFS and TOF traces. The trace data is written to the SYSPRINT data set.

The traces are activated by switches in the MVS EXEC statement PARM parameter:

- If the **first** character of the value of the PARM parameter is **T**, the **MFS** trace is activated.
- If the **second** character of the value of the PARM parameter is **T**, the **TOF** trace is activated.

For example:

```
EXEC PGM=IMRSQLP,PARM='T'     (starts the SQL processor with an MFS trace)
EXEC PGM=IMRSQLP,PARM=' T'    (starts the SQL processor with a TOF trace)
EXEC PGM=IMRSQLP,PARM='TT'    (starts the SQL processor with both an MFS and TOF trace)
```

Terminating the SQL Processor

Enter this command as a reply to the WTOR operator message IMR700A or IMR701A:

TERM **STOP**	

The SQL processor finishes the current operations and terminates.

Disconnecting the SQL Processor

Enter this command as a reply to the WTOR operator message IMR700A (SQL processor is connected).:

DISC	

The SQL processor enters a wait state. No polling for work is done.

The SQL processor is reactivated by the CONN command.

Connecting the SQL Processor

Enter this command as a reply to the WTOR operator message IMR701A (SQL processor is running but disconnected):

CONN	

The SQL processor restarts polling for work according to the polling frequencies specified in the parameter module IMRPRM.

Part 3. Administering Traffic Reconciliation

Chapter 6. Migrating from Version 3.3

When migrating a Version 3.3 Traffic Reconciliation installation to Version 4, consider the following:

- All customization tables must be regenerated. These are IMRPRM, scanner control tables, the table filter table, and the query table.

  A number of detailed changes have been made to the customization macro parameters but, since V3.3 tables can be processed by the V4 macros, conversion to the new and changed parameters is not necessary. Nevertheless, it is recommended.
- When regenerating IMRPRM, assembler errors are possible due to the improved consistency and completeness checks in macro IMRPARM. Your IMRPRM module must be changed accordingly. The following parameters are no longer accepted:
  - QNAME
  - SRVSQLID
  - USER
- For DB2 tables:
  - V3.3 PTFs UQ14845 and UQ14846 added column EKEY to the Telex Link and MERVA Link HDR tables. If you have not applied these PTFs, you will need to:
    - Unload the tables concerned
    - Add the EKEY column to the tables
    - Reload the tables, adding the PKEY column value also to the EKEY column
  - S.W.I.F.T. tables containing an MIR or MOR must be expanded to hold the MIR or MOR subfields. You need to:
    - Use the SQL ALTER statement to define the new columns. Samples IMRAFIN and IMRAGPA contain the necessary statements.
    - If you want to access existing data using the new columns you need to update the modified tables by copying the MIR and MOR values to the new columns. Samples IMRUFIN and IMRUGPA contain the necessary SQL UPDATE statements.

      Alternatively, if the tables contain large amounts of data, you may prefer to use the unload-reload method.
  - Column MSG_CATEGORIE in the MERVA Link HDR tables must be renamed to MSG_CATEGORY. Any queries or programs you have written that refer to this column must be changed.
- The event record formats have changed, so the flip-flop must be completely processed by V3.3 before migration. After MERVA ESA V3.3 has terminated for the last time, from a CICS or IMS terminal manually initiate the insertion transaction twice to ensure all events from both flip-flop data sets are inserted to DB2.
- The V3.3 queue management batch utility (IMRQMGB) has been incorporated into the MERVA ESA base product and renamed DSLSQB. If you use this utility, you must change the program name and the STEPLIB in your JCL.
- Users of MCB-based queries should be informed of the changed PF-key usage.

Converting IMRPRM

As mentioned in the previous section, detailed changes to your V3.3 IMRPRM parameter module may be necessary because of the improved checking. In addition a number of simplifications have been incorporated.

To convert a V3.3 IMRPRM to V4:

- Replace each occurrence of macros **IMRPxxx** by **IMRPARM**. Note that all **TYPE=EVENT** entries must immediately follow the corresponding **TYPE=MONITOR** specification.
- Remove the **QNAME** parameter. The DSLPRM **NAME** parameter is used instead of **QNAME**.
- Remove the **SRVSQLID** parameter. You are recommended to bind the IMRSQLP plan with **DYNAMICRULES=BIND**.
- Change **FFTRESH** to **FFTHRESH**.
- Replace the **MSGID** and **FORMID** parameters in the telex monitor specification by **TLXFLDS**.
- In the queue and telex monitor specifications change **CONTROL** to **SCANTAB**.
- In the MERVA Link specifications change any **MCB** keyword to **MTYPE**.
- The first subparameter of the **EVENT** parameter in a MERVA Link **TYPE=EVENT** specification is now mandatory.
- Remove the **USER** parameter. You must specify both the MERVA Link MFS exit and the Traffic Reconciliation MFS exit to extract MERVA Link events in the MERVA Link partner table.
- Remove the **APPC** parameter.

Converting Scanner Control Tables

Like IMRPRM the use of the scanner control macros IMRQUS and IMRTXS has been simplified to one macro, IMRSCAN.

To convert V3.3 scanner control tables to V4 it is only necessary to:

- Replace all occurrences of **IMRQUS** by **IMRSCAN**.
- Replace all occurrences of **IMRTXS** by **IMRSCAN**.
- Change any **LIN** parameter to **LINE**.

To improve readability, you can change any EBCDIC characters in hexadecimal **BEG** or **END** tags to character notation.

Converting the Table Filter

The IMRFLD macro has been replaced in V4 by macro IMRTFL with **TYPE=FIELD**.

To convert your V3.3 table filter to V4:

- Change each occurrence of **IMRFLD** to **IMRTFL**.
- Change the **TYPE=** specification for each table from **ENTRY** to **TABLE**.

Chapter 7. Calculating Space Requirements for Traffic Reconciliation

The amount of disk space you require for your DB2 tables, and the flip-flop data sets if you use them, depends on which events you monitor, the volume of traffic you need to support, and for how long you want to retain data in the database.

Space Requirements for DB2 Tables

The Traffic Reconciliation table space creation DDL reserves space for approximately 1000 events. This is intended to be sufficient for a preliminary test system, and to allow simple scaling-up to production database sizes.

Estimate, for each monitor, how many events will be recorded per day, and for how many days the data is to be held in DB2. Then adjust the PRIQTY specifications in the Traffic Reconciliation DDL accordingly.

The space required by one row in each table is shown in column **Recl** in "Appendix A. DB2 Table Overview" on page 95. The number of rows that fit in a DB2 page is:

```
(PageSize - 22) / (Recl + 2)
```

VARCHAR Columns

Note, however, that these **Recl** values include the maximum length for rows containing VARCHAR columns. The average length will normally be less, often much less. For example, the queue monitor field tables specify a length of 256 for field data. However, the MSGTRACE field, which will often be recorded, is only 28 bytes long, and other fields will often be even smaller.

You can change the size of the VARCHAR columns in the various tables used for message documentation and event field extraction. If you do, the **MAXLEN** value in the corresponding table filter entry must be changed accordingly.

The size of these columns depends on how you use Traffic Reconciliation. You are recommended to use the default values supplied by IBM to begin with and to make changes in the light of experience, or to accomodate special requirements.

Space Requirements for the Flip-Flop

If you are using these data sets, their approximate size needs to be calculated before they can be allocated. This again varies greatly depending on which events are being monitored, and how large typical messages and field extraction strings are. Normally, you will want to make the flip-flop data sets large enough to hold at least as much data as is produced during one full day of operation.

Product-Sensitive Programming Interface

The format of flip-flop event records is defined in the copybook IMRFILE.

End of Product-Sensitive Programming Interface

The following table summarizes the sizes of event records for each monitor. You may want to use different values for the average message string and field string sizes when you calculate your space requirements. The values are approximate.

Table 2. Event Record Sizes (in Bytes)

Monitor	Common Prefix	Specific Prefix	Msg. String	Fld. Strings	Total
SWIFT (input)	50	45	1500	-	1600
SWIFT (output)	50	175	1500	-	1750
Queue	50	200	-	100	350
Journal	50	50	500	-	600
Telex	50	50	5000	200	5300
MLINK	50	80	1500	400	2100
MQI	50	300	1500	250	2100

Chapter 8. Customizing

IMRPRM Parameters Module

IMRPRM is the primary Traffic Reconciliation customization module. In the same way as the MERVA ESA parameters module, DSLPRM is built using assembler macro DSLPARM, IMRPRM is built using assembler macro IMRPARM. In this section macro IMRPARM and its parameters are described.

Name	Operator	Operands

	IMRPARM	TYPE={CONTROL}
		{MONITOR}
		{EVENT}
		{FINAL}
		TYPE=CONTROL parameters (general):
		[DB2={*cccc*}]
		{DB2 }
		[DUMQUE={*cccccccc*}]
		{IMRI }
		[PLANEXTP={*cccccccc*}]
		{DSLNUC }
		[PLANINSB={*cccccccc*}]
		{IMRINSP }
		[PLANJRNX={*cccccccc*}]
		{IMRJRNX }
		[PLANMLKR={*cccccccc*}]
		{EKATPI1 }
		[PLANQUEX={*cccccccc*}]
		{IMRQUEX }
		[PLANRDU={*cccccccc*}]
		{IMRRDU }
		[PLANSQLP={*cccccccc*}]
		{IMRSQLP }
		[PLANSWFX={*cccccccc*}]
		{IMRSWFX }
		[SYNCDB2={NO }]
		{YES}
		[TFILTER={*cccccccc*}]
		{IMRTFLT }
		[TIV=({*mm*,*ss*})]
		{ 5, 0}
		[TQUERY={*cccccccc*}]
		{IMRQRYT }

Name	Operator	Operands

		TYPE=CONTROL parameters for flip-flop data sets:
		[COMMIT={nn}]
		{ 5}
		[FFTHRESH={nnn}]
		{100}
		[RNAME=cccccccc]
		TYPE=CONTROL parameters for the SQL processor:
		[SRVERR={cccccccc}]
		{IMRSQLE }
		[SRVFULL=({nn,nn})]
		{ 0, 0}
		[SRVIN={cccccccc}]
		{IMRSQLI }
		[SRVSTOP={nn}]
		{ 0}
		[SRVTMP={cccccccc}]
		{IMRSQLT }
		[SRVWAIT=({nn,nn})]
		{ 2,15}
		TYPE=MONITOR parameters:
		NAME={SWIFT }
		{QUEUE }
		{JOURNAL}
		{MLINK }
		{TELEX }
		{MQI }
		[ASP=cccccccc]
		[PROCESS=cccccccc]
		[SCANTAB=cccccccc]
		[SYNCDB2={NO }]
		{YES}
		TYPE=EVENT parameters for the SWIFT Link monitor:
		APPL={GPA}
		{FIN}
		[SCOPE={INPUT }]
		{OUTPUT}
		{BOTH }
		[TRNINHDR={NO }]
		{YES}

Name	Operator	Operands

		TYPE=EVENT parameters for the queue monitor:
		NAME={cccccccc}
		
		[DELFLDS=(cccccccc,c)]
		
		[MSGTRACE={YES}]
		{NO }
		
		[PUTFLDS=(cccccccc,c)]
		
		[SCANTAB=cccccccc]
		
		[SCOPE={PUT }]
		{DELETE}
		{BOTH }
		
		TYPE=EVENT parameters for the journal monitor:
		JRNID={xx}
		{xx-yy}
		
		TYPE=EVENT parameters for the telex monitor:
		SCOPE={INPUT }
		{OUTPUT}
		
		[TLXFLDS=(cccccccc,c)]
		
		TYPE=EVENT parameters for the MERVA Link monitor:
		{S}
		{O}
		EVENT=({C},{MSG})
		{I} {ACK}
		{R} {*}
		{V}
		{*}
		
		[CTL=({MTYPE},{cccccccc},{c},cccccccc)]
		{ƀƀƀƀƀƀƀƀ} {*}
		{NET },{cccccccc} {#}
		{DSLEXIT }
		
		[DOC=({MTYPE},{cccccccc},{c})]
		{ƀƀƀƀƀƀƀƀ} {*}
		{NET },{cccccccc} {#}
		{DSLEXIT }
		
		[FLD=({MTYPE},{cccccccc},{c},cccccccc)]
		{ƀƀƀƀƀƀƀƀ} {*}
		{NET },{cccccccc} {#}
		{DSLEXIT }
		
		TYPE=EVENT parameters for the MQI monitor:
		EVENT=({DATAGRAM}{,REQUEST}{,REPLY}{,REPORT}{,ALL})
		
		[FLD=({MTYPE},{cccccccc},{c},cccccccc)]
		{ƀƀƀƀƀƀƀƀ} {*}
		{NET },{cccccccc}
		{DSLEXIT }

IMRPARM TYPE=CONTROL

The **TYPE=CONTROL** statement starts the parameters module and contains monitor-independent parameters.

General TYPE=CONTROL Parameters

DB2
The subsystem name of your DB2 system.

DUMQUE
The MERVA ESA function associated with the event insertion transaction.

PLANEXTP
The extraction process plan name. The name is only required if you are using synchronous DB2 event insertion in a MERVA batch nucleus. If you are also using DB2 for MERVA queue management, the plan must include the queue management DBRMs.

PLANINSB
The plan name for the insertion process (IMRINSP) when running as a batch program.

PLANJRNX
The journal monitor extraction process plan name. The name is only required if you have defined synchronous DB2 insertion for the journal monitor, and the journal extraction service (IMRJRNX) is running as a subtask of a MERVA batch nucleus.

PLANMLKR
The plan name for the MERVA Link APPC/MVS receive transaction program, EKAPTI1. This entry is only required if you are using the APPC/MVS transaction scheduler to initiate the MERVA Link receive transaction for IMS connections.

PLANQUEX
The queue monitor extraction process plan name. The name is only required if you have defined synchronous DB2 insertion for the queue monitor, and the queue extraction service (IMRQUEX) is running as a subtask of a MERVA batch nucleus.

PLANRDU
The plan name for the delete utility.

PLANSQLP
The SQL processor (IMRSQLP) plan name.

PLANSWFX
The S.W.I.F.T. monitor extraction process plan name. The name is only required if you have defined synchronous DB2 insertion for the S.W.I.F.T. monitor, and the S.W.I.F.T. extraction service (IMRSWFX) is running as a subtask of a MERVA batch nucleus.

SYNCDB2
Events for any particular monitor can either be inserted directly into the DB2 tables, or written to the flip-flop data sets. This is controlled by the **SYNCDB2** parameter of the **TYPE=MONITOR** specification, the default value for which is determined by this parameter:

- **SYNCDB2=NO** causes all monitors without a SYNCDB2 specification to write events to the flip-flop for subsequent, asynchronous insertion into the DB2 database by the event insertion transaction.
- **SYNCDB2=YES** causes all monitors without a SYNCDB2 specification to bypass the flip-flop and insert events directly into the DB2 database (synchronous DB2 insertion).

TFILTER
The name of the table filter.

TIV
The time interval (minutes, seconds). Defines how often the insertion transaction controller, IMRICON, will reinitiate the event insertion transaction IMRINSP.

TQUERY The name of the query table. The query table defines all Traffic Reconciliation MCB-based queries.

TYPE=CONTROL Parameters for Flip-Flop Data Sets

COMMIT The default value for both of the following:

- During extraction, the number of flip-flop records written before a commit is performed. For the extraction process, a commit means updating the control record in the IMRCTL data set, and performing a VSAM temporary close of the current event data set, which causes VSAM to complete outstanding I/O and update the catalog.
- During insertion, the number of flip-flop records read before a commit is performed. During the insertion process, the IMRCTL table is updated, an SQL COMMIT is performed, and the flip-flop control record is updated.

FFTHRESH The flip-flop threshold percentage. Event extraction issues a warning message (IMR031W) when this percentage of space in the flip-flop (both data sets taken together) is taken up by events not yet processed by the insertion process. A second message (IMR032W) is issued when the percentage drops back below this threshhold.

RNAME A name that, combined with the **NAME** parameter from the MERVA parameters module DSLPRM, can be used to uniquely identify the flip-flop as an MVS serially reusable resource (MVS ENQ/DEQ).

TYPE=CONTROL Parameters for the SQL Processor

SRVERR MERVA queue to which the SQL processor routes erroneous and rejected requests.

SRVFULL Defines the percentages of the fullness of the MERVA queue data set (QDS) and large message cluster (LMC) beyond which query lists and retrievals are no longer generated by the SQL processor. If the QDS or LMC are more full than the percentages specified here, the SQL processor stops processing requests and rejects new requests in order not to overload the system. The first value is the percentage for the QDS; the second for the LMC. A value of zero skips the load checking of the corresponding storage.

SRVIN MERVA queue that defines the input queue for the SQL processor. MCB-based query requests should be routed to this queue, which is inspected periodically for work by the SQL processor.

SRVSTOP Specifies how often the SQL processor attempts to connect to MERVA. After this number of unsuccessful attempts the SQL processor terminates. The retry frequency is defined by the second parameter of **SRVWAIT**.

SRVTMP MERVA queue that serves as a temporary queue for the SQL processor.

SRVWAIT Defines the frequency in seconds with which the SQL processor polls the **SRVIN** queue. The first number specifies the normal polling frequency. In case of environment errors (for example, MERVA not available), the SQL processor polls according to the second number.

IMRPARM TYPE=MONITOR

Each Traffic Reconciliation monitor is defined by a **TYPE=MONITOR** entry followed by one or more **TYPE=EVENT** entries. There can be only one monitor for SWIFT Link, Telex Link, the journal, and queue management. For MERVA Link and MQI there can be one monitor per MERVA Link ASP or MERVA-MQI attachment process. The following parameters can be specified:

ASP
Required if a MERVA Link monitor is being defined. It specifies the name of the MERVA Link ASP that is to be monitored. A MERVA Link ASP is defined in the MERVA Link partner table.

Note: The MLINK monitor program is an MFS exit with number 7089. In the MERVA Link partner table (EKAPT), for each ASP that is to be monitored, specify this exit number as the second value of the MFSEXIT parameter, for example:

- If no other MFS exits are used, specify **MFSEXIT=(,7089)**
- If a USE connection MFS exit is used, specify **MFSEXIT=(7133,7089)**

NAME
The name of the monitor. It must be **QUEUE**, **SWIFT**, **JOURNAL**, **TELEX**, **MLINK**, or **MQI**. For MERVA Link and MQI more than one monitor can be defined.

PROCESS
Required if a MERVA-MQI attachment monitor is being defined. It specifies the MQI process to be monitored. An MQI process is defined in the MERVA-MQI attachment process table.

SCANTAB
The name of the Telex scanner control table, or of the default scanner control table for queue events.

SYNCDB2
Flip-flop file usage. **SYNCDB2=NO** indicates that event extraction is to write this monitor’s events to the event flip-flop for later, asynchronous, insertion into the DB2 database by the event insertion transaction.

SYNCDB2=YES causes the event extraction process to bypass the flip-flop and insert the events directly into the DB2 database (synchronous DB2 insertion).

The default is the value defined by the **SYNCDB2** parameter on the **TYPE=CONTROL** specification, or **SYNCDB2=NO** if that was omitted.

When monitoring queue management events generated by a program or transaction using direct DB2 queue management (DSLPRM **SDDB2** parameter) this parameter is ignored and events are inserted into DB2 synchronously.

IMRPARM TYPE=EVENT

TYPE=EVENT specifications for a particular monitor must immediately follow the TYPE=MONITOR specification for that monitor. At least one **TYPE=EVENT** specification is required for each monitor specified.

Queue Monitor TYPE=EVENT Parameters

The queue monitor **TYPE=EVENT** parameters are described here. You can specify any number of queue monitor events.

NAME
The name of the MERVA ESA queue that is to be monitored. The name can include wildcard characters:

- * To represent any character string
- % To represent any single character

DELFLDS
When a message is deleted, moved, or routed from this queue, it is formatted with this message identifier and format identifier for the purpose of field extraction. The first subparameter specifies a message identifier (DSLMTT MTYPE parameter); the second specifies a format identifier.

If **DELFLDS** is omitted (or specified as **DELFLDS=**), no fields are inserted into the QUEDFLD table.

If **DELFLDS** is specified but either of the subparameters is omitted, the default message ID is the message ID of the message in the TOF, and the default format ID is the first format ID in the MCB identified by this message ID. For example, if you specify **DELFLDS=(,W)**, the message is mapped into S.W.I.F.T. net format.

MSGTRACE
If **YES** is specified, the last data area of the MSGTRACE field of the queue element is inserted into the appropriate field table (QUEPFLD or QUEDFLD) with a FIELD_ID of **MSGTRACE**.

The MSGTRACE field is described in *MERVA for ESA V4 Concepts and Components*.

PUTFLDS
Similar to **DELFLDS**, but for queue PUT events. When a message is put into the queue it will be formatted with this message identifier (DSLMTT MTYPE parameter) and format identifier for the purpose of field extraction. The first subparameter specifies a message identifier (DSLMTT MTYPE parameter); the second specifies a format identifier.

If **PUTFLDS** is omitted (or specified as **PUTFLDS=**), no fields will be inserted into the QUEPFLD table.

By default, the message ID is the message ID of the message in the TOF, and the format ID is the first format ID in the MCB identified by the message ID. For example, when you code **PUTFLDS=(,W)**, the message in the TOF is mapped into S.W.I.F.T. net format.

SCANTAB
The name of the scanner control table to be used for field extraction. If omitted and field extraction is specified, that is, either **PUTFLDS** or **DELFLDS** is specified, the scanner control table specified in the **TYPE=MONITOR** specification is used.

SCOPE
Specifies the queue events to be monitored:

PUT
A PUT event occurs every time MERVA ESA queue management writes a queue element to the specified queue. This can be as a result of message creation, routing, or a REQUEUE command.

DELETE
A DELETE event occurs every time queue management deletes a message from the specified queue. This can be when a message is routed to another queue, or as the result of a DELETE or REQUEUE command.

BOTH
Both PUT and DELETE events are to be monitored.

S.W.I.F.T. Monitor TYPE=EVENT Parameters

The SWIFT Link monitor **TYPE=EVENT** parameters are:

APPL The S.W.I.F.T. application that is to be monitored (**FIN** or **GPA**). This parameter is required.

SCOPE Specifies the S.W.I.F.T. events to be monitored:

INPUT
Messages being sent to the S.W.I.F.T. network

OUTPUT
Messages received from the network

BOTH Both input and output S.W.I.F.T. messages

TRNINHDR Controls the use of column U_MUR in the FINIHDR and FINOHDR tables.

TRNINHDR=NO causes the MUR from the S.W.I.F.T. user header to be inserted into column U_MUR. If there is no MUR, the TRN, field 20, is inserted into U_MUR.

TRNINHDR=YES forces the TRN to be put into column U_MUR, regardless of the presence of a MUR.

The table filter further qualifies which S.W.I.F.T. messages are captured to DB2.

Journal Monitor TYPE=EVENT Parameters

Any number of journal monitor **TYPE=EVENT** entries can be defined. There is only one parameter:

JRNID Specifies a hexadecimal journal record identifier to be monitored, or a range of identifiers (nn-mm) that are to be monitored. See the appendix of *MERVA for ESA V4 Concepts and Components* for a list of valid journal identifiers.

Telex Monitor TYPE=EVENT Parameters

The Telex Link monitor **TYPE=EVENT** parameters are:

SCOPE Defines this entry as applying to either telex input (telexes sent from MERVA ESA to the telex network) or to telex output (telexes received from the telex network by MERVA ESA).

TLXFLDS A telex message will be formatted with this message identifier (DSLMTT MTYPE parameter) and format identifier for the purpose of field extraction. When the telex event is inserted into the DB2 tables the scanner control table (**SCANTAB**) determines which fields are extracted from this message string. The parameter is optional.

By default the message ID is the message ID of the message, and the format ID is the first format ID in the MCB identified by the message ID.

MLINK Monitor TYPE=EVENT Parameters

The use of these parameters is discussed in “DOC, CTL, and FLD Parameters” on page 61. The table filter further qualifies which MERVA Link messages and fields are captured to DB2.

EVENT EVENT selects the type of the MERVA Link event to be monitored.

The first subparameter is mandatory and must be one of the following:

S	Ready-to-send message
O	Outgoing message
C	Confirmed message

I Incoming application message
R Status report
V Recovered or rerouted message
***** Any of the events listed above

These values correspond to the MERVA Link user exit function values, which are described in the section of *MERVA for ESA V4 Advanced MERVA Link* that describes user exit support in MERVA Link, and in the sample MERVA Link exit EKAMU010.

The second subparameter must be one of:

MSG MERVA Link application messages are to be monitored.

ACK MERVA Link status reports are to be monitored.

***** Both application messages and status reports are to be monitored.

DOC The DOC specification defines how the MERVA Link messages are represented in the DB2 MLINK DOC tables. If omitted no row is written to the DOC table.

CTL The CTL specification defines which control fields are inserted into the DB2 MLINK CTL tables, and how they are inserted. If omitted, no rows are inserted into the CTL tables.

FLD The FLD specification defines which message fields are inserted into the DB2 MLINK FLD tables, and how they are inserted. If omitted, no rows are inserted into the FLD tables.

DOC, CTL, and FLD Parameters: To extract a message documentation string and to define message strings for field extraction MCBs must be defined. The necessary specifications are provided in the **DOC=**, **CTL=**, and **FLD=** parameters.

If the first subparameter is **MTYPE**, the second subparameter specifies the message identifier (DSLMTT MTYPE parameter) of the MCB to be used. If the second subparameter is omitted, the message identifier is taken from the TOF field named in the TOF NLEXIT field.

Otherwise, when the first subparameter is **NET**, the second subparameter specifies the TOF field containing the message identifier. Default is TOF field DSLEXIT.

The third subparameter identifies which format ID in the MCB is to be used to format the required message string. It can be specified explicitly, or indirectly by specifying * or **#**. An * indicates that the first character of the message identifier is the format ID, a **#** indicates that the format ID is to be taken from TOF field EKANETID.

The fourth subparameter of the **CTL=** and **FLD=** parameters specifies the scanner control table to be used by Traffic Reconciliation when scanning the formatted message string for fields to be inserted to the CTL or FLD tables.

Example:

```
CTL=(MTYPE,IMRMLKC,S,IMRMLKT)          (1)
FLD=(NET,DSLEXIT,W,IMRMLKT)            (2)
```

Notes:

1. Format S in the MCB identified by MTYPE IMRMLKC in the MERVA message type table (DSLMTT) is used to create a message string for field extraction.
2. Format W in the MCB identified by the message identifier in TOF field DSLEXIT is used to map the string for extracting fields to a MERVA Link FLD table.

The scanner control table in both cases is IMRMLKT.

MQI Monitor TYPE=EVENT Parameters

EVENT The MQI event to be monitored. This can be one or more of **DATAGRAM**, **REQUEST**, **REPLY**, and **REPORT**, or **ALL** to indicate all of these.

FLD The FLD parameters define how the MQI message TOF is to be mapped for field extraction. The specification is the same as for the MERVA Link **FLD** specification, except that a **#** is not allowed for the third subparameter. Refer to “DOC, CTL, and FLD Parameters” on page 61.

The table filter further qualifies which MQI messages and fields are captured to DB2.

IMRPARM TYPE=FINAL

This statement must be the last statement in the IMRPRM module.

Table Filter

The table filter controls which DB2 tables are filled by Traffic Reconciliation. It is a further data selection step, qualifying the insertion process, in contrast to IMRPRM and scanner control tables, which qualify the extraction of data from MERVA.

Only those DB2 tables defined in the table filter can be filled by the insertion process. Tables not defined will remain empty. If a parent table is omitted from the filter, then any subordinate tables will also not be filled. This is because the necessary parent key, for example DKEY or AKEY, will not be available.

Additionally, field insertion for SWIFT, MLINK, and MQI events can be influenced. For S.W.I.F.T. events, the table filter is the only place where extraction and insertion of message fields can be controlled.

The following example shows an extract from the sample table filter IMRTFLT:

```
IMRTFLT  IMRTFL TYPE=INITIAL
         IMRTFL TYPE=TABLE,NAME=FINIDOC,MAXLEN=11000
         IMRTFL TYPE=TABLE,NAME=FINODOC,MAXLEN=11000,SCOPE=(SYS,USE)
         IMRTFL TYPE=TABLE,NAME=FINISRC
         IMRTFL TYPE=TABLE,NAME=GPAISRC
         ...
         IMRTFL TYPE=TABLE,NAME=FINIREF
         IMRTFL TYPE=FIELD,MSGTYPE=???,FIELDS=(20,21)
         IMRTFL TYPE=TABLE,NAME=FINOREF
         IMRTFL TYPE=FIELD,MSGTYPE=???,FIELDS=(3??)
         ...
         IMRTFL TYPE=TABLE,NAME=FINOTRL,MAXLEN=32,SCOPE=(PDE,PDM)
         ...
         IMRTFL TYPE=TABLE,NAME=MLKIFLD,MAXLEN=256
IMRTFL TYPE=FIELD,MSGTYPE=S100,FIELDS=(SWC1C)
IMRTFL TYPE=FIELD,MSGTYPE=S200,FIELDS=(3??,5??,EKA?????)
...
IMRTFL TYPE=FINAL
```

IMRTFL Macro

The table filter is defined using the IMRTFL macro. The IMRTFL macro has the following parameters:

Name	Operator	Operands
	IMRTFL	**TYPE**={**INITIAL**} {**TABLE**} {**FIELD**} {**FINAL**} **TYPE**=**TABLE** parameters: [**INSERT**={**YES**}] {**NO** } [**MAXLEN**={*nnnnn*}] {**32704**} **NAME**=*cccccccc* [**SCOPE**=(*ccc*[,*ccc*])] **TYPE**=**FIELD** parameters: **MSGTYPE**=*cccccccc* [**FIELDS**=(*cccccccc*[,*cccccccc*])]

TYPE=TABLE parameters

A **TYPE=TABLE** entry is required for each DB2 table into which rows are to be inserted. If the table contains an AKEY column or a DKEY column, then a **TYPE=TABLE** entry is also required for the corresponding parent table.

INSERT
YES, the default, specifies that data is to be inserted into this table. Specifying **NO**, or omitting a **TYPE=TABLE** entry for a table, inhibits data insertion into the table, and into any subordinate tables.

MAXLEN
For a DOC table, a field table, a TRL table, and for the JRNPUT table, the length of the VARCHAR column in the table must be specified with this parameter. Data longer than **MAXLEN** will be truncated before insertion. If **MAXLEN** is larger than the actual maximum length of the VARCHAR column, an SQL error can occur. **MAXLEN** cannot be larger than 32704.

NAME
Specifies the name of the DB2 table.

Note: You cannot prevent insertions into the S.W.I.F.T. MER, TAR, and SRC tables.

SCOPE
The purpose of **SCOPE** is to restrict the S.W.I.F.T. events that are to be recorded. The parameter applies only to the documentation tables FINxDOC and GPAxDOC, and to the trailer tables FINxTRL and GPAxTRL.

The **SCOPE** operand consists of a list of comma-separated values enclosed in parentheses. Event categories omitted from the list are not recorded. But if the **SCOPE** parameter is omitted, insertion is not restricted.

For the documentation tables, the following values are possible:

SRV	S.W.I.F.T. service messages (service identifier other than 01 and 21)
SYS	S.W.I.F.T. system messages (service identifier 01 and not FIN, and FIN message types less than 100)
USE	S.W.I.F.T. user-to-user messages (FIN message types >= 100)
ACK	S.W.I.F.T. acknowledgments (service identifier 21)

Note: If an event is precluded from insertion into a S.W.I.F.T. DOC table, that event will not be recorded at all, since the DOC table is the root table for S.W.I.F.T. events. (This does not apply to the MER, TAR, and SRC tables, because they are not subordinate to the DOC table.)

Possible values for a TRL table are any S.W.I.F.T. trailer identifier (**PDE**, **PDM**, for example). A **SCOPE** parameter associated with a TRL table specifies the trailers that are to be recorded in the TRL table. Trailers omitted from the list are not recorded. But again, if no **SCOPE** parameter is specified, all trailers are recorded.

TYPE=FIELD Parameters

A **TYPE=TABLE** entry for some field tables may be followed by one or more **TYPE=FIELD** entries to limit insertions to that field table.

The following field tables can be qualified in this way:

FINxFLD	S.W.I.F.T. FIN field table
GPAxFLD	S.W.I.F.T. GPA field table
FINxAMN	S.W.I.F.T. FIN amount table
FINxREF	S.W.I.F.T. FIN reference field table
MLKxFLD	MLINK field table
MLKxCTL	MLINK control field table
MQIxFLD	MQI field table

If no **TYPE=FIELD** entry is coded:

- For S.W.I.F.T. events, all qualifying fields extracted from the message are inserted into the field table (assuming the event is inserted into the parent tables). For FLD and REF tables, these are all fields in S.W.I.F.T. message blocks 3 and 4. For the AMN table, these are all amount fields found in block 4.
- For events other than S.W.I.F.T. events, any fields extracted by the field scanner as controlled by a scanner control table are inserted into the field table (assuming the event is inserted into the parent tables).

Insertion to other field tables, for example F32, cannot be limited in this way. If a **TYPE=TABLE** entry is present, all qualifying fields are inserted.

Parameters for a **TYPE=FIELD** entry are:

MSGTYPE Defines the message type to which this entry applies. The message type can contain one or more **?** characters as wildcards representing any single character. Only fields of messages of the specified type are considered for insertion.

For S.W.I.F.T. tables the message type is the 3-character S.W.I.F.T. message type from the S.W.I.F.T. application header.

For MLINK and MQI tables MSGTYPE identifies the MERVA ESA message identifier (MTYPE parameter in the message type table, DSLMTTT) of length 8.

FIELDS Specifies one or more tags of fields to be inserted into the table identified in the preceding **TYPE=TABLE** entry. Each tag can contain one or more **?** characters as wildcards representing any single character.

For S.W.I.F.T. tables, up to three characters can be specified. The tags must be S.W.I.F.T. field tags (including options), or a 3-character subblock number.

For MLINK and MQI tables the tags are the 8-character field identifiers generated by the scanner from the FID parameters in the scanner control table.

Example of a Table Filter

The following is part of a table filter. The description that follows refers to the line numbers in parentheses.

```
IMRTFL NAME=FINIDOC,SCOPE=(USE,SYS)              (1)
IMRTFL NAME=FINIHDR                              (2)

IMRTFL NAME=FINIFLD                              (3)
IMRTFL MSGTYPE=(20?),FIELDS=(5??)                (4)
IMRTFL MSGTYPE=(01?),FIELDS=(106,107,108)        (5)

IMRTFL NAME=FINIAMN                              (6)
IMRTFL MSGTYPE=(100),FIELDS=(3??)                (7)
IMRTFL MSGTYPE=(20?),FIELDS=(3??)                (8)
IMRTFL MSGTYPE=(400),FIELDS=(3??)                (9)

IMRTFL NAME=FINIREF                             (10)
IMRTFL MSGTYPE=(???),FIELDS=(20?)               (11)

IMRTFL NAME=MLKIFLD                             (12)
IMRTFL MSGTYPE=(S100),FIELDS=(SW20)             (13)
```

All FIN input messages except service messages and ISN ACKs are recorded (1). Since message fields are to be recorded, the header table, the parent table of the field tables, must be specified (2). The following fields from blocks 3 and 4 of FIN input messages are recorded:

- All fields of category 5 from message types 20x (4)
- Subblocks 106, 107, and 108 (related MIR, MOR, MUR) from message types 01x (5)
- All amount fields of category 3 from message types 100 (7), 20x (8), and 400 (9)
- All fields 20 (TRN) from every message (11)

Fields specified by (4, 5) are inserted in the general field table FINIFLD (3), amount fields in the amount table FINIAMN (6), and reference fields in the reference table FINIREF (10).

Fields assigned to field ID SW20 by the MLINK scanner (using IMRSCAN macro **FID** parameter) are inserted in the MLINK MLKIFLD table from MERVA Link input messages with a message type of S100.

Fields Required by G9x Commands

The G9x end user commands retrieve S.W.I.F.T. messages from the S.W.I.F.T. monitor database.

For a message to be retrieved by these commands the FINxDOC and FINxHDR tables must be filled, and the FINxREF tables must contain the TRN of the message.

Field Extraction

Generally, when Traffic Reconciliation gets control from MERVA to process an event, it receives a message TOF (the message in tokenized form) as well as a message buffer (the message in network format). While the message buffer can serve as source for event documentation, the TOF provides access to individual message fields.

The exceptions to this are the two monitors not associated with a network, the queue and journal monitors. Queue events have no message buffer, only a TOF, so there is no DOC table for queue events. Journal events have no TOF, only the journal record, so you cannot extract fields from a journal event.

To capture fields from a TOF, Traffic Reconciliation requires two steps:

1. Map the required fields from the TOF to a string using an MCB (MERVA message control block).

   The resultant string can contain any number of fields in the form **begin-tag, data, end-tag**.
2. Select fields from this string under control of a scanner control table and insert them into the DB2 tables.

   A scanner control table specifies the tags of fields to be selected and the name with which the field is to be stored in DB2.

This mechanism is not used for S.W.I.F.T. field extraction because S.W.I.F.T. messages already have well-known string formats and field tags. Fields are selected from the S.W.I.F.T. network format message and selection is controlled completely by specifications in the table filter.

Table filter specifications can also be used to further influence insertion of fields selected by the scanner process for MLINK and MQI events.

Field String Mapping

To generate a string for the scanner an MCB must be prepared. Here is the sample MCB, IMRQFLD, provided by Traffic Reconciliation for queue events:

```
         TITLE  'IMRQFLD - QUEUE
MONITOR FIELD MCB SAMPLE'
IMRQFLD  DSLLMCB
***********************************************************************
* PUT EVENT
***********************************************************************
NETP     DSLLDEV   TYPE=NET,ID=P,SEP=''
         DSLLNFLD  TAG=':20:',FLD=SW20,VFIRST=YES,FSEP='::'
         DSLLNFLD  TAG=':32:',FLD=SW32,VFIRST=YES,FSEP='::'
         DSLLNFLD  TAG=':50:',FLD=SW50,VFIRST=YES,FSEP='::'
         DSLLNFLD  TAG=':59:',FLD=SW59,VFIRST=YES,FSEP='::'
***********************************************************************
* DELETE EVENT
***********************************************************************
```

```
NETD     DSLLDEV   TYPE=NET,ID=D,SEP=''
         DSLLNFLD  TAG=':20:',FLD=SW20,VFIRST=YES,FSEP='::'
         DSLLGEN
         END
```

An entry is necessary in the Traffic Reconciliation parameters module IMRPRM to instruct the queue monitor to use this MCB. For example, to extract S.W.I.F.T. fields 20, 32, 50, and 59 from every message passing through a MERVA message verification function you could add an entry like this to your queue monitor specifications:

```
IMRPARM NAME=L*VE0,SCOPE=BOTH,MSGTRACE=YES,                 *
     PUTFLDS=(IMRQFLD,P),DELFLDS=(IMRQFLD,P)
```

Each time a queue element is written to or deleted from one of the verification queues (L1VE0, L2VE0, or L3VE0), a string is generated using format ID **P** in the MCB identified by MTYPE IMRQFLD. (In this case, both the MTYPE and the MCB are named IMRQFLD.) If any of the fields specified in the MCB (**FLD** parameter) exist in the message, they are added to the string preceded by the corresponding tag, and followed by **::**.

Similarly, fields can be extracted from an outgoing Telex Link message with the following line in IMRPRM:

```
IMRPARM SCOPE=INPUT,TLXFLDS=(IMRTLXM,I)
```

IMRTLXM is a sample MCB provided by Traffic Reconciliation.

The MERVA Link monitor supports two sets of field tables:

- CTL tables, intended for MERVA Link control fields
- FLD tables, intended for fields from the application message being transferred by MERVA Link

Consequently two string mappings for fields can be generated:

```
IMRPARM TYPE=MONITOR,NAME=MLINK,ASP=A1I
IMRPARM EVENT=(O,MSG),                                        *
     CTL=(MTYPE,IMRMLKC,O,IMRFMTT),                           *
     FLD=(MTYPE,IMRFMTC,O,IMRFMTT),                           *
     DOC=(NET,DSLEXIT,W)
```

The MCBs IMRMLKC and IMRFMTC are provided by Traffic Reconciliation.

Mapping of the field string for MQI events is similar, but there is no separate CTL field string:

```
IMRPARM TYPE=MONITOR,NAME=MQI,PROCESS=SPROC1
IMRPARM EVENT=ALL,FLD=(NET,DSLEXIT,W,IMRMQIT)
```

For more information on writing MCBs, refer to the *MERVA for ESA V4 Customization Guide*.

Scanner Control Tables (IMRSCAN Macro)

The second step in extracting fields from messages is the scanning of the string created by the MCB, under the control of a scanner control table.

A scanner control table identifies the beginning and end tags of each field to be selected from the MCB-generated string, and the name under which the field is to be stored in the DB2 field table.

The name is stored in column FIELD_ID of the appropriate FLD table, and the field value is stored in column FIELD_DATA. In all cases the FIELD_DATA column is defined as a VARCHAR column. Traffic Reconciliation takes the length of the column from the **MAXLEN** parameter in the corresponding table filter entry. If the value in the field is longer than **MAXLEN**, it is truncated. (The column names in the MLINK CTL field tables are CTL_ID and CTL_DATA.)

The sample queue monitor scanner control table, IMRQUET, looks like this:

```
         TITLE 'Queue Scanner Control Table Sample'
IMRQUET  IMRSCAN TYPE=INITIAL
***********************************************************************
* This entry extracts fields mapped by the sample MCB IMRQFLD:
***********************************************************************
         IMRSCAN BEG=C':??:',END=C'::',FID=C'SW??',CTL=SKIP
         IMRSCAN BEG=(X'0D25',C':???:'),END=(X'0D25',C':'),FID=C'???',   *
               CTL=STAY
         IMRSCAN BEG=(X'0D25',C':???:'),END=(X'0D25',C'-'),FID=C'???',   *
               CTL=SKIP
         IMRSCAN BEG=(X'0D25',C':??:'),END=(X'0D25',C':'),FID=C'??',     *
               CTL=STAY
         IMRSCAN BEG=(X'0D25',C':??:'),END=(X'0D25',C'-'),FID=C'??',     *
               CTL=SKIP
         IMRSCAN TYPE=FINAL
```

The first line after the **TYPE=INITIAL** line selects all the fields extracted by the sample MCB IMRQFLD used in the previous section. It uses wildcard characters to find any tags of the appropriate pattern and to form the field name inserted into the FIELD_ID column. IMRQUET also shows how any generic block-4 fields can be selected from a S.W.I.F.T. message formatted in the default S.W.I.F.T. net format.

IMRSCAN Macro

A scanner control table is defined with the IMRSCAN macro. Any number of field selection specifications can be defined (without a **TYPE=** parameter).

The field selection parameters are:

Name	Operator	Operands

	IMRSCAN	[**TYPE**={**INITIAL**}]
		{**FINAL** }
		
		BEG={**C'***cccccccc***'**}
		{**X'***cccccccc***'**}
		{(**C'***cccccccc***',X'***cccccccc***'**[,...])}
		
		[**CTL**={**SKIP**}]
		{**STAY**}
		
		END={**C'***cccccccc***'**}
		{**X'***cccccccc***'**}
		{(**C'***cccccccc***',X'***cccccccc***'**[,...])}
		
		FID=C'*cccccccc***'**
		
		[**LINE**=(*m*,*n*)]
		
		[**SECTION**=({**INPUT** },{**TELEX**})]
		{**OUTPUT**},{**USER** }

The meaning of these IMRSCAN parameters is as follows:

TYPE The first line in a scanner control table must contain only the

TYPE=INITIAL parameter, the last line must contain only **TYPE=FINAL**. Omit **TYPE=** for other entries.

BEG Defines the beginning tag of a field. Use assembler character notation or hexadecimal notation without any further attributes. Both character and hexadecimal notation can be combined within parentheses to form complex tags. Tags can be of any length. You can use one or more **?** characters as wildcards in any position except the first to represent any single character.

CTL Controls how the scanner, after it has found and processed a field, continues its scan for the next field. **SKIP**, the default, means that the scanner continues scanning at the character following the end tag. **STAY** means that it continues scanning at the first character of the end tag. The end tag of a field can thus form (part of) the beginning tag of the next field.

END Defines the ending tag of a field. It can be specified in exactly the same way as the **BEG** tag.

FID Defines the name that will be inserted into the FIELD_ID column to identify this field. Use assembler character notation. The name can be up to 8 characters long.

Note: For Telex fields only the leftmost three characters are used.

You can specify one or more question marks that correspond to those specified for the **BEG** parameter. These will be replaced by the corresponding characters in the matching beginning tag.

LINE For the telex scanner table only, specifies two line numbers defining a range of lines in a telex to be selected as a field. A line is any sequence of characters terminated by a CRLF sequence (X'0D25'). The **FID** parameter defines the name to be used to identify the field in the DB2 table.

The first line number must not be greater than the second.

Instead of a line number an asterisk representing the last line of the telex can be specified. If an asterisk is used, it can be combined with a minus sign and a number to identify the *n*th line before the last line:

```
IMRSCAN LINE=(*-5,*),FID=C'END'
```

SECTION For the telex scanner table only. Four different buffers can be presented to the telex scanner. For each buffer a separate section in the scanner control table can be defined:

- **SECTION=(INPUT,USER)** is used for the buffer generated by the telex MCB for input telexes (outgoing telexes).
- **SECTION=(OUTPUT,USER)** is used for the buffer generated by the telex MCB for output telexes (incoming telexes).
- **SECTION=(INPUT,TELEX)** is used for the buffer containing the actual input telex APDU.
- **SECTION=(OUTPUT,TELEX)** is used for the buffer containing the actual output telex APDU.

When defining the telex scanner control table, a field specification, that is, a **LINE** range or a **BEG–END** specification, must be preceded by a **SECTION** statement.

Example MCBs and Scanner Control Tables

A number of example MCBs and tables are provided by Traffic Reconciliation. The MCBs are:

IMRFMTC	MLINK FMT control fields
IMRFMTD	MLINK FMT acknowledgment as documentation string
IMRMLKC	MLINK control fields
IMRQFLD	Queue event field
IMRTLXM	Host-based Telex fields
IMRTX2C	Workstation-based Telex fields
IMRUSEC	MLINK USE control fields
IMRUSEF	MLINK USE application fields

The sample scanner control tables are:

IMRFMTT	MLINK FMT application
IMRMLKT	MLINK S.W.I.F.T. message events
IMRMQIT	MQI S.W.I.F.T. message events
IMRQUET	Queue event
IMRTLXT	Host-based Telex event
IMRTX2T	Workstation-based Telex event
IMRUSET	MLINK USE events

MCB-Based Queries

MCB-based queries are queries integrated into the MERVA ESA end user terminal environment.

When adding or modifying MCB-based queries the following points need to be considered:

IMRQRY For any MCB-based query an entry is required in the query table, IMRQRYT, using the **IMRQRY** macro.

Query MCB Each query must be defined by an MCB.

Queues and Routing
Both queries and their results are messages that are routed or stored in MERVA queues.

List MCB List queries return an SQL result table as a MERVA message. An MCB is required to display the result table.

Message Retrievals
Retrieval queries return MERVA application messages, for example S.W.I.F.T. messages. MCBs for these messages already exist.

Permissions Each user can be assigned a Traffic Reconciliation user class in their user file entry. Users can only initiate those queries that have been defined with the user's class in the query table.

Queue Keys Composite key fields are defined for queues containing queries and their results.

PF Keys As in any MERVA end user panel the PF keys in the QRY selection and query entry panels can be modified.

Enabling the DELALL command
The DELALL command cannot be freely used.

MCB-based queries are normally entered from the query selection panel (QRY), which is defined in copybook IMRFNTTC, part of the MERVA function table.

They can also be entered from any data entry function if:

- The query is defined with MTGEN=YES in the message type table (see copybook IMRMTTTC).
- The user is permitted access to the message type of the query, that is, the message type. For example, Q001 is one of the user's permitted message types.
- Routing is changed to route the completed query to the SQL processor input queue. The results of the query are routed according to the definitions in the query table.

For more information on MCBs, refer to the *MERVA for ESA V4 Customization Guide*.

IMRQRY Macro

All MCB-based queries must be defined in the query table. Traffic Reconciliation provides a sample table with the name IMRQRYT. If the table is given another name, the name must be specified in the parameters module IMRPRM, parameter **TQUERY**.

A query table is defined by a series of IMRQRY macros. Here is part of the standard table:

```
IMRQRY TYPE=LIST,
      MTYPE=Q001,
      DESCR='List of FIN Input/Output Messages',
      CLASS=(1,4,3),
      LIST=(L001,L001LIN,53),
      TARGET=(R,IMRSQL0),
      MAX=18,
      INIT=((Q001ORDR,'D'))
IMRQRY TYPE=RETRIEVAL,
      MTYPE=Q005,
      DESCR='SWIFT Input Messages + ACK/NAK',
      CLASS=(1,4,3),
      MAP=(,W,MSGACK),
      TARGET=(Q,QRYRTV0),
      MAX=17
```

The IMRQRY macro is defined as follows:

Name	Operator	Operands
	IMRQRY	**CLASS**=(*n*[,...]) **DESCR**='*ccccccc*' **INIT**=((*cccccccc*,'*ccccccc*')[,...]) **LIST**=(*cccccccc*,*cccccccc*,*cccccccc*) **MAP**=(*cccccccc*,*c*[,*cccccccc*]...) **MAX**=*n* **MTYPE**=*cccccccc* **TARGET**=({**Q**},*cccccccc*) {**R**} **TYPE**={**INITIAL** } {**FINAL** } {**LIST** } {**RETRIEVAL**}

CLASS Lists the user classes that are permitted to run this query. A user class must be assigned to each user who wants to run queries. This is done by coding a user class number in the **Recon User Class** field in the MERVA user-file panel. Classes must be in the range 1 to 254.

DESCR Description of the query, up to 54 characters. This text is displayed on the query selection panel next to the query's message type.

INIT Specifies a list of *field, value* pairs defining TOF fields and the default value to which the field should be set when the query is selected. The value must be enclosed in quotes.

The defaults are set by the Traffic Reconciliation default setting exit 4001. Each field named should have default setting exit 4001 defined either in its FDT entry or in the query MCB.

LIST For a **TYPE=LIST** query, this defines the message type of the MCB to be used to format the result list (subparameter 1), and the TOF field into which each row of the result is written (subparameter 2).

The concatenation of each column in a row of the result list is written to a data area of the TOF field. The third subparameter specifies the maximum length that will be written to these data areas. If the concatenation of columns is longer, it will be truncated.

MAP For message retrieval queries, **TYPE=RETRIEVAL**, this defines how the retrieved messages are to be formatted.

The first column of each result row is mapped as a MERVA message. The first subparameter specifies the message identifier of the MCB to be used. If omitted, MERVA inspects the message and determines its type. The second subparameter specifies the format ID to be used.

The remaining subparameters name TOF fields into which the remaining columns of each row are written, at nesting level 0. The insertion of TOF fields terminates when the list of field names is exhausted, or there are no more columns.

MAX
Defines the maximum number of rows, whether lists or retrievals, to be retrieved by the query.

MTYPE
Defines the message identifier of the query. There must be a corresponding entry in the MERVA message type table.

TARGET
Defines how the queue elements containing the query results are to be added to the MERVA message queuing system.

If Q is specified as the first subparameter, the queue elements are written directly to the MERVA queue named by the second subparameter.

If R is specified as the first subparameter, the queue elements are routed using the routing table of the MERVA queue named by the second subparameter.

TYPE
The type of IMRQRY entry. It must be one of the following:

INITIAL
The first IMRQRY entry in the query table must specify **TYPE=INITIAL** only.

FINAL
The last IMRQRY entry in the query table must specify **TYPE=FINAL** only.

LIST
The query generates a general result table of columns and rows, a list. An MCB to display the list must be defined.

RETRIEVAL
The query retrieves MERVA application messages from the database. The MCB to be used to map the messages must be specified.

Query MCBs

A query MCB defines:

- The panel that is displayed to the end user when the query is selected in the query selection function. DSLLDEV TYPE=SCREEN.
- The SQL SELECT statement, the actual query. DSLLDEV TYPE=NET.

Here is MCB IMRQ005, which defines Q005, one of the sample MCB-based queries distributed with Traffic Reconciliation:

```
        TITLE  'IMRQ005 - SWIFT FIN Input Retrievals SAMPLE'
        COPY IMRMSQL                   <==== IMR Global Defs       ( 1)
        COPY DSLCOLOR
IMRQ005  DSLLMCB                                                   ( 2)
**********************************************************************
* Message
**********************************************************************
MESSAGE  DSLLDEV  TYPE=MESSAGE                                     ( 3)
        COPY IMRMMSG                   <==== IMR system fields     ( 4)
Q005MTY  DSLLMFLD MAND=YES             Message Type                ( 5)
Q005DES  DSLLMFLD MAND=YES             Destination                 ( 5)
Q005FIX  DSLLMFLD                      Field id X                  ( 5)
Q005FDX  DSLLMFLD                      Field Data X                ( 5)
Q005FIY  DSLLMFLD                      Field id Y                  ( 5)
Q005FDY  DSLLMFLD                      Field Data Y                ( 5)
Q005FIZ  DSLLMFLD                      Field id Z                  ( 5)
Q005FDZ  DSLLMFLD                      Field Data Z                ( 5)
**********************************************************************
* Screen
```

```
***********************************************************************
SCREEN   DSLLDEV  TYPE=SCREEN,ID=E                                 ( 6)
         COPY IMRMRSP                  <==== IMR response          ( 7)
         DSLLDFLD '----------------------------------------------------*
               ----------------Description',POS=(NEXT,2)
         DSLLDFLD 'This query retrieves SWIFT FIN input messages togeth*
               er with their ACK/NAK.',POS=(NEXT+1,2)
         DSLLDFLD 'The messages and the response are routed via the MER*
               VA queue',POS=(NEXT,2)
         DSLLDFLD FLD=IMR01TAR,POS=(,NEXT),LENGTH=8,PROT=YES       ( 8)
         DSLLDFLD '.',POS=(,NEXT)
         DSLLDFLD 'No more than',POS=(NEXT,2)
         DSLLDFLD FLD=IMR01LIM,POS=(,NEXT),LENGTH=5                ( 8)
         DSLLDFLD 'messages are fetched from the database.',           *
               POS=(,NEXT)
         DSLLDFLD 'You are restricted to retrieve messages sent from:',*
               POS=(NEXT,2)
         DSLLDFLD FLD=IMR01OR8,POS=(,NEXT),LENGTH=8,PROT=YES       ( 8)
         DSLLDFLD '----------------------------------------------------*
               ------------------Selection',POS=(NEXT+1,2)
         DSLLDFLD 'Message type         *:',POS=(NEXT+1,2)
         DSLLDFLD FLD=Q005MTY,POS=(,NEXT),LENGTH=3
         DSLLDFLD 'Destination pattern  *:',POS=(NEXT,2)
         DSLLDFLD FLD=Q005DES,POS=(,NEXT),LENGTH=8,EDIT=4021       ( 9)
         DSLLDFLD 'Tag/data pattern      :',POS=(NEXT+1,2)
         DSLLDFLD FLD=Q005FIX,POS=(,NEXT),LENGTH=3,EDIT=4021       ( 9)
         DSLLDFLD FLD=Q005FDX,POS=(,NEXT),LENGTH=40,EDIT=4021      ( 9)
         DSLLDFLD 'Tag/data pattern      :',POS=(NEXT,2)
         DSLLDFLD FLD=Q005FIY,POS=(,NEXT),LENGTH=3,EDIT=4021       ( 9)
         DSLLDFLD FLD=Q005FDY,POS=(,NEXT),LENGTH=40,EDIT=4021      ( 9)
         DSLLDFLD 'Tag/data pattern      :',POS=(NEXT,2)
         DSLLDFLD FLD=Q005FIZ,POS=(,NEXT),LENGTH=3,EDIT=4021       ( 9)
         DSLLDFLD FLD=Q005FDZ,POS=(,NEXT),LENGTH=40,EDIT=4021      ( 9)
         DSLLDFLD 'Messages to skip      :',POS=(NEXT+1,2)
         DSLLDFLD FLD=IMR01SKP,POS=(,NEXT),LENGTH=5                ( 8)
         DSLLDFLD '----------------------------------------------------*
               ---------------End of Query',POS=(NEXT+1,2)
***********************************************************************
* SELECT statement
***********************************************************************
NETQ     DSLLDEV  TYPE=NET,ID=Q,SEP=''                             (10)
***********************************************************************
* Declaration of host variables
***********************************************************************
IMR01OR8 DCL CHAR                                                  (11)
Q005MTY  DCL CHAR                                                  (11)
Q005DES  DCL CHAR                                                  (11)
Q005FIX  DCL CHAR                                                  (11)
Q005FDX  DCL CHAR                                                  (11)
Q005FIY  DCL CHAR                                                  (11)
Q005FDY  DCL CHAR                                                  (11)
Q005FIZ  DCL CHAR                                                  (11)
Q005FDZ  DCL CHAR                                                  (11)
***********************************************************************
* Result Table
***********************************************************************
         SQL 'SELECT M.APDU, A.APDU, M.PKEY'                       (12)
***********************************************************************
* FROM clause
***********************************************************************
         SQL 'FROM FINIDOC M, FINIDOC A,'
         SQL '  FINIHDR H, FINIACK I'
         JE  (Q005FIX,''),Q100                                     (13)
         JE  (Q005FDX,''),Q100
         SQL '  ,FINIFLD X'
Q100     JE  (Q005FIY,''),Q110
         JE  (Q005FDY,''),Q110
         SQL '  ,FINIFLD Y'
Q110     JE  (Q005FIZ,''),Q200
         JE  (Q005FDZ,''),Q200
         SQL '  ,FINIFLD Z'
**********************************************************************
* WHERE clause
**********************************************************************
Q200     SQL 'WHERE M.PKEY = H.DKEY'
         SQL 'AND   A.PKEY = I.DKEY'
         SQL 'AND H.B_BANK || H.B_COUNTRY || H.B_LOCATION =',:IMR01OR8
         SQL 'AND   H.B_SEQUENCE = I.B_SEQUENCE'
         SQL 'AND   H.B_SESSION = I.B_SESSION'
         SQL 'AND   H.RETRIEVAL = '' '''
         SQL 'AND   H.A_MSGTYPE = ',:Q005MTY
         SQL 'AND   H.A_DESTIN LIKE ',:Q005DES
         JE  (Q005FIX,''),Q210
         JE  (Q005FDX,''),Q210
         SQL 'AND   X.AKEY = H.PKEY'
         SQL 'AND   X.FIELD_ID LIKE ',:Q005FIX
         SQL 'AND   X.FIELD_DATA LIKE ',:Q005FDX
Q210     JE  (Q005FIY,''),Q220
         JE  (Q005FDY,''),Q220
         SQL 'AND   Y.AKEY = H.PKEY'
         SQL 'AND   Y.FIELD_ID LIKE ',:Q005FIY
         SQL 'AND   Y.FIELD_DATA LIKE ',:Q005FDY
Q220     JE  (Q005FIZ,''),Q300
         JE  (Q005FDZ,''),Q300
         SQL 'AND   Z.AKEY = H.PKEY'
         SQL 'AND   Z.FIELD_ID LIKE ',:Q005FIZ
         SQL 'AND   Z.FIELD_DATA LIKE ',:Q005FDZ
**********************************************************************
* ORDER clause
**********************************************************************
Q300     SQL 'ORDER BY 3 DESC FOR FETCH ONLY'
**********************************************************************
* Printers
**********************************************************************
HARDCOPY DSLLDEV  TYPE=HARDCOPY,ID=E,LIKE=SCREEN                   (14)
SYSP     DSLLDEV  TYPE=SYSP,ID=E,LIKE=SCREEN                       (14)
         DSLLGEN
         END
```

Notes:

1. The query MCB must start with a COPY statement for IMRMSQL, which defines some macros that simplify coding of the SQL statement. These macros are DCL, SQL, JMP, JE, JNE, and NOP and are described later.
2. The name of the MCB is defined by the DSLLMCB macro.
3. This part defines TOF fields to be included in the query message.
4. The copy book IMRMMSG defines mandatory Traffic Reconciliation TOF system fields for query messages:

   IMR01TYP Defines the type of the query, either L (list query) or R (retrieval query). The default is set according to the definition in the query table.

   IMR01LIM Maximum number of rows that can be returned by this query. The default value is the MAX value from the query table.

   IMR01PUT Defines whether the generated list or retrieval messages are to be put directly into a queue (Q) or routed (R). The default is the **TARGET** parameter value from the query table.

IMR01TAR Defines the target queue for direct putting or indirect routing. The default is the **TARGET** parameter value from the query table.

IMR01SRC Defines the source of the query. The default is the MERVA function or queue where the query is entered.

IMR01ORI Defines the origin identification of the user submitting the query. The default is the origin identification defined in the user file entry. The subfield IMR01OR8 defined in the copy book IMRFDTTC specifies the first 8 characters of the origin. In most cases this is the origin S.W.I.F.T. destination of the end user.

Additionally the subfields IMR01OBC, IMR01OCC, and IMR01OLC define the bank code, country code, and location code.

IMR01SKP This number of rows in the result table is skipped before list or retrieval processing starts. Together with the IMR01LIM field this allows a subset of a result table to be retrieved or listed.

This parameter is optional. Default is zero.

5. These are additional TOF fields for the query.
6. This starts the definition of the query entry panel. Instructions and parameters are displayed on the screen.
7. The query message becomes a response message after IMRSQLP has processed the query. Copy book IMRMRSP defines the standard response panel. The response is not to be confused with the result of the SQL SELECT: a retrieval or a list.
8. The Traffic Reconciliation system fields can be displayed and changed by the end user.
9. The distributed edit routine 4021 (IMRME021) translates the common wildcard characters ***** and **?** into the DB2 specific wildcard characters **%** and **_**.

   In order to further enhance the end user dialog, MFS exit routines can be coded to set defaults, and check or edit input data.
10. This statement starts the definition of the SQL statement.
11. The attributes of the input fields used in the SQL statement are declared using the DCL macro. A field can be either a character (CHAR) or numeric (NUM) field. The label must specify a TOF field. Up to 16 variables may be declared.

    The SQL processor does not support FLOAT, GRAPHIC, VARGRAPHIC, or LONG VARGRAPHIC data types.
12. The various parts of the SQL statement can be defined using the SQL macro. The first positional parameter is any static text string enclosed in quotes. A second parameter specifying a variable in the form of a previously defined TOF field can follow. The variable must be preceded by a colon.

    The value of the variable in the TOF is inserted into the SQL statement when MFS formats the net format of the query message.
13. Conditional logic can be defined in an MCB using the DSLLCOND macro. This can be simplified by using these macros:

    JMP Jumps unconditionally. JMP can be preceded by a label.

```
         JMP L001
         ...
L001     NOP
```

JE Causes a jump if the specified TOF field contains the specified data. The data must be enclosed in quotes. JE can be preceded by a label.

```
         JE  (Q001XYZ,'ABC'),L001
         ...
L001     JE  (Q001FGH,'123'),L002
```

JNE Causes a jump if the specified TOF field does not contain the specified data. The data must be enclosed in quotes. The syntax is the same as for JE.

NOP Defines a label. It has no other effect.

```
L001     NOP
```

14. This specifies the representation of the query MCB on hardcopy and system printers.

Exits

Four MFS exits are provided by Traffic Reconciliation. These can be useful when writing your own MCB-based queries. The exits are:

IMRMD001 Default setting exit providing access to parameters in the query table entry

IMRMC011 Checking exit for parameters in a query table

IMRME021 Edit exit to convert Traffic Reconciliation wildcard characters to their SQL equivalents

IMRMS031 Separation exit to construct the keys for the query response, retrieval, and list queues

For a description of MFS exits, refer to the section that describes MFS exit program classes in the *MERVA for ESA V4 System Programming Guide*.

IMRMD001 Default Setting Exit

Any MCB-based query must be defined in an entry in the query table. In the MCB defining a query you can use this default setting exit to extract values specified in some of the parameters in the query table entry. To activate the exit specify **DEFAULT=4001** on a field specification.

The necessary field specifications are provided in copybook IMRMMSG, which should be included in the query MCB. The fields concerned and the corresponding parameters in a query table entry are:

IMRQRYL **MTYPE=,** subfield IMRQRYLQ, and **DESC=**, subfield IMRQRYLD. These fields are defined in the field definition table (see IMRFDTTC), not in copybook IMRMMSG.

IMR01LIM **MAX=,** the maximum number of messages or rows to be retrieved.

IMR01PUT **TARGET=** subparameter 1, the queuing method to be used for result messages (put or route).

IMR01TAR **TARGET=** subparameter 2, the queue for result message routing.

IMR01TYP **TYPE=,** the type of query: list or retrieval.

The following fields do not reference the query table, but are also set by IMRMD001:

IMR01SRC The function where the query is entered.

IMR01ORI The origin ID from the user file record of the user initiating the query. The subfields IMR01OBC, IMR01OCC, and IMR01OLC are defined for this field to give access to the bank, country, and location codes in the origin ID (see IMRFDTTC).

Additionally, IMRMD001 handles the default setting for any fields specified in a query table **INIT=** parameter. To activate this, not only must a field and its default value be specified in the **INIT=** parameter, but the field definition must also be available to MFS (in a DSLLMFLD definition or in the FDT) and specify **DEFAULT=4001**.

For example, the sample query Q001 defines the field Q001ORDR, sorting order, in this way. The field is defined directly in the MCB (IMRQ001):

```
Q001ORDR DSLLMFLD MAND=YES,DEFAULT=4001
```

and in the query table entry for Q001:

```
         IMRQRY TYPE=LIST,MTYPE=Q001,                                  *
               DESCR='List of FIN Input/Output Messages',              *
               CLASS=(1,4,3),                                          *
               LIST=(L001,L001LIN,53),                                 *
               TARGET=(R,IMRSQLO),                                     *
               MAX=18,                                                 *
               INIT=((Q001ORDR,'D'))
```

When Q001 is selected, IMRMD001 presets the sorting order to **D**.

IMRMC011 Checking Exit

The checking exit IMRMC011 checks the following fields for validity. The exit is activated by specifying **CHECK=4011** in the field definition. These fields are defined in the FDT (see IMRFDTTC), and the use of the exit is specified in copybook IMRMMSG.

IMR01LIM The maximum number of messages or rows to be retrieved. The value is checked for 1 to 5 digits.

IMR01PUT The queuing method to be used for result messages. The value can contain only **Q** or **R**.

IMR01TAR The queue for result message routing. IMRMC011 checks that the queue is defined in the MERVA function table.

IMR01TYP The type of query. The value must be **L** or **R**.

IMR01SKP This field is used in a number of the sample queries to specify how many rows in the result table are to be skipped. The value is checked for 1 to 5 digits.

If a check fails, an error message is displayed in the query input panel.

IMRME021 Wildcard Edit Exit

An MFS edit exit is used to edit a TOF field for display on a screen, and to deedit screen input for storage in the TOF. The edit exit IMRME021 can be used to transform the Traffic Reconciliation wildcard characters ***** and **?** to the corresponding substitution characters % and _ used in the SQL **LIKE** predicate. Specify **EDIT=4021** on a field specification to activate editing and deediting for terminal display and input.

For example, in sample query Q001 the TRN and correspondent can contain wildcard substitution characters. The part of the **TYPE=SCREEN** definition concerned looks like this (from MCB IMRQ001):

```
         DSLLDFLD 'Pattern for TRN/MUR                        :',      *
               POS=(NEXT,2)
         DSLLDFLD FLD=Q001TRNM,POS=(,NEXT),LENGTH=16,EDIT=4021
         DSLLDFLD 'Pattern for Correspondent                  :',      *
               POS=(NEXT,2)
         DSLLDFLD FLD=Q001CORR,POS=(,NEXT),LENGTH=8,EDIT=4021
```

The **EDIT=4021** specification means that when an end user enters a TRN or correspondent value containing a * or **?** the field is stored in the message TOF with these characters changed (deedited) to % and _. The characters are changed back (edited) to * or **?** when these TOF fields are displayed on a screen.

Editing is not performed when a message is displayed in NOPROMPT format, so when the generated SQL statement is displayed using the **NOPROMPT** command or PF11, the SQL substitution characters are visible.

Omit **EDIT=4021** if you prefer to enter the SQL substitution characters directly.

IMRMS031 Key Separation Exit

Separation exit IMRMS031 generates key fields for use in queues to which responses, lists, and retrievals are routed after processing by IMRSQLP. These keys are discussed in “Queue Keys” on page 83.

List MCBs

The purpose of a list MCB is to define how the information returned by a list query is presented to the end user.

The SQL processor, IMRSQLP, processes list queries in the following steps:

1. The query is checked and the SQL statement executed.
2. A TOF is initialized with the list MCB specified by the **LIST** parameter in the query table entry for this list query.
3. Rows are fetched from the SQL result table until no more rows are available or the maximum number is reached (**MAX** parameter in the query table).
4. The columns of each row are concatenated and written, possibly truncated, to the list message as one data area of the TOF field specified in the **LIST** parameter of the query table entry.
5. The system field IMR01LST is added to the list message TOF (see “Routing of Queries and Results” on page 81).
6. The system fields IMR01QRY and IMR01LOG are added to the TOF of the query message. The query is now called a response.
7. The list message and the query response message are written to a MERVA queue according to the **TARGET** specification in the query table entry for the list query.

Here is IMRL001, the list MCB for query Q001 (numbers in parentheses refer to the notes that follow):

```
         COPY DSLCOLOR
IMRL001  DSLLMCB
MESSAGE  DSLLDEV  TYPE=MESSAGE
L001LIN  DSLLMFLD                                                   (1)
SCREEN   DSLLDEV  TYPE=SCREEN,ID=E
         DSLLDFLD 'Date  Time  Typ Corresp. P T MUR/TRN          SeqNum*
                MervaUMR',POS=(NEXT,2),DISP=HIGH
         DSLLDFLD '----------------------------------------------------*
               ---------------------------',POS=(NEXT,2)
         DSLLUNIT DACNT=(1,32700),COMPRES=NO
         DSLLCOND O1=(TEST=L001LIN),EQ=YES,O2=',GOTO=SL900
         DSLLDFLD FLD=L001DATE,POS=(NEXT,2),LENGTH=5,PROT=YES       (2)
         DSLLDFLD FLD=L001TIME,POS=(,NEXT),LENGTH=5,PROT=YES
         DSLLDFLD FLD=L001MTYP,POS=(,NEXT),LENGTH=3,PROT=YES
         DSLLDFLD FLD=L001CORR,POS=(,NEXT),LENGTH=8,PROT=YES
         DSLLDFLD FLD=L001PRIO,POS=(,NEXT),LENGTH=1,PROT=YES
         DSLLDFLD FLD=L001MTID,POS=(,NEXT),LENGTH=1,PROT=YES
         DSLLDFLD FLD=L001TRNM,POS=(,NEXT),LENGTH=16,PROT=YES
         DSLLDFLD FLD=L001BSEQ,POS=(,NEXT),LENGTH=6,PROT=YES
         DSLLDFLD FLD=L001UMRM,POS=(,NEXT),LENGTH=8,PROT=YES
         DSLLCOND LINES=3,GOTO=SL100
         DSLLCOND GOTO=SL190
SL100    DSLLDFLD '----------------------------------------------------*
               -----------------------More',POS=(NEXT,2)
         DSLLCOND PAGE=NEW
         DSLLDFLD 'Date  Time  Typ Corresp. P T MUR/TRN          SeqNum*
                MervaUMR',POS=(NEXT,2),DISP=HIGH
         DSLLDFLD '----------------------------------------------------*
               ---------------------------',POS=(NEXT,2)
SL190    DSLLCOND
         DSLLUEND
SL900    DSLLDFLD '----------------------------------------------------*
               ----------------End of List',POS=(NEXT,2)
HARDCOPY DSLLDEV  TYPE=HARDCOPY,ID=E,LIKE=SCREEN
SYSP     DSLLDEV  TYPE=SYSP,ID=E,LIKE=SCREEN
         DSLLGEN
         END
```

Notes:

1. Defines the list field. Each data area of the list field receives the concatenated columns of one row fetched from DB2.
2. It can simplify the display of columns if the field is divided into subfields. This requires definition of the field and subfields in the MERVA FDT, and a separation exit. Separation can be performed by either standard or user-written separation routines.

   L001LIN is defined in the FDT as follows (see copy book IMRFDTTC). 901 is the standard MERVA separation exit for separating fixed length subfields:

   ```
   L001LIN  DSLLFLD  LENGTH=(0,53,V),DAMAX=32767,SEPR=901
   L001DATE DSLLSUBF LENGTH=5,OFFSET=0         Date MM-DD
   L001TIME DSLLSUBF LENGTH=5,OFFSET=5         Time HH-MM
   L001MTYP DSLLSUBF LENGTH=3,OFFSET=10        Message type
   L001CORR DSLLSUBF LENGTH=8,OFFSET=13        Correspondent
   L001PRIO DSLLSUBF LENGTH=1,OFFSET=21        Priority
   L001MTID DSLLSUBF LENGTH=1,OFFSET=22        MUR/TRN indicator
   L001TRNM DSLLSUBF LENGTH=16,OFFSET=23       TRN/MUR
   L001BSEQ DSLLSUBF LENGTH=6,OFFSET=39        Sequence number (basic)
   L001UMRM DSLLSUBF LENGTH=(0,8,V),OFFSET=45  Merva UMR sequence number
   ```

Message Retrievals

During processing of a retrieval query the SQL processor builds retrieval messages according to specifications in the query table.

The SQL processor, IMRSQLP, processes retrieval queries in the following way:

1. The query is checked and the SQL statement executed.
2. Rows are fetched from the result table until no more rows are available or the maximum number is reached (**MAX** parameter in the query table). Each row is converted into one MERVA message.
3. The first column of each row is mapped into a TOF under the control of the first and second subparameters of the **MAP** parameter in the query table entry. The first subparameter specifies a message identifier used for mapping, and the

second specifies a format ID. If the message identifier is not specified, MERVA message type determination derives the appropriate MCB from the column's contents. This is the usual method for S.W.I.F.T. messages.

4. All other columns are written at nesting level 0 into the TOF fields named in the next subparameters of the **MAP** parameter. This is done until no more columns are defined or no more TOF field names are specified.
5. Field IMR01RTV is written to nesting level 0 (see "Routing of Queries and Results").
6. The retrievals are written to MERVA queues according to the **TARGET** specifications in the query table entry.
7. Fields IMR01QRY and IMR01LOG are written to the query message. The query is now called a response.
8. The response is also written to a MERVA queue according to the **TARGET** specifications in the query table entry.

Here is an example of how messages are created. Assume the **MAP** parameter in a query table entry specifies:

```
              MAP=(,W,MSGACK),
```

and the query specifies:

```
         SQL 'SELECT M.APDU, A.APDU, M.PKEY'
         ..
Q300     SQL 'ORDER BY 3 DESC FOR FETCH ONLY'
```

Then column M.APDU, presumably a S.W.I.F.T. message in a DOC table, is mapped into an empty TOF using the S.W.I.F.T. MCB determined by MERVA message type determination and format ID W. Additionally, column A.APDU is written to TOF field MSGACK at nesting level 0. The third column is only used for sorting the SQL result table, it is not added to the TOF.

Routing of Queries and Results

After processing by the SQL processor lists, retrievals, and query responses are routed under control of the **TARGET** parameter in the query table. IMRSQLP inserts system fields into the TOF at nesting level zero to assist routing. It also inserts operator messages into field IMR01LOG indicating the success of the query.

The TOF field IMR01QRY is attached to the query message. This turns it into a query response message. The structure of IMR01QRY is defined in the MERVA FDT:

```
IMR01QRY DSLLFLD  LENGTH=(,,U),SEPR=901  Query record
IMR01QDT DSLLSUBF LENGTH=6,OFFSET=0      Date
IMR01QTI DSLLSUBF LENGTH=6,OFFSET=6      Time
IMR01QQR DSLLSUBF LENGTH=8,OFFSET=12     Query name
IMR01QID DSLLSUBF LENGTH=10,OFFSET=20    Query ID
IMR01QUS DSLLSUBF LENGTH=8,OFFSET=30     User
IMR01QPG DSLLSUBF LENGTH=8,OFFSET=38     Program name
IMR01QRC DSLLSUBF LENGTH=2,OFFSET=46     Return code (see IMR741E and IMR761E)
IMR01QSR DSLLSUBF LENGTH=8,OFFSET=48     Source queue
```

The TOF field IMR01LST is attached to list messages. The structure is defined in the MERVA FDT:

```
IMR01LST DSLLFLD  LENGTH=(,,U),SEPR=901  List record
IMR01LDT DSLLSUBF LENGTH=6,OFFSET=0      Date
IMR01LTI DSLLSUBF LENGTH=6,OFFSET=6      Time
IMR01LQR DSLLSUBF LENGTH=8,OFFSET=12     Query name
IMR01LID DSLLSUBF LENGTH=10,OFFSET=20    Query ID
```

```
IMR01LUS DSLLSUBF LENGTH=8,OFFSET=30     User
IMR01LPG DSLLSUBF LENGTH=8,OFFSET=38     Program name
IMR01LRC DSLLSUBF LENGTH=2,OFFSET=46     Return code (see IMR741E)
IMR01LLS DSLLSUBF LENGTH=8,OFFSET=48     List name
IMR01LSR DSLLSUBF LENGTH=8,OFFSET=60     Source queue
```

The TOF field IMR01RTV is attached to retrieved application messages. The structure is defined in the MERVA FDT:

```
IMR01RTV DSLLFLD  LENGTH=(,,U),SEPR=901  Retrieval record
IMR01RDT DSLLSUBF LENGTH=6,OFFSET=0      Date
IMR01RTI DSLLSUBF LENGTH=6,OFFSET=6      Time
IMR01RQR DSLLSUBF LENGTH=8,OFFSET=12     Query name
IMR01RID DSLLSUBF LENGTH=10,OFFSET=20    Query ID
IMR01RUS DSLLSUBF LENGTH=8,OFFSET=30     User
IMR01RPG DSLLSUBF LENGTH=8,OFFSET=38     Program name
IMR01RRC DSLLSUBF LENGTH=2,OFFSET=46     Return code (always 0)
IMR01RRT DSLLSUBF LENGTH=8,OFFSET=48     Retrieval name
IMR01RSR DSLLSUBF LENGTH=8,OFFSET=56     Source queue
```

In the query table supplied by Traffic Reconciliation, IMRQRYT, the **TARGET** parameter specifies routing under control of function IMRSQLO. This function is associated with routing table IMRSQLRT (see IMRFNTTC). Refer to this routing table for examples of the use of these fields.

DELALL and CLEAR commands are not routed but simply deleted.

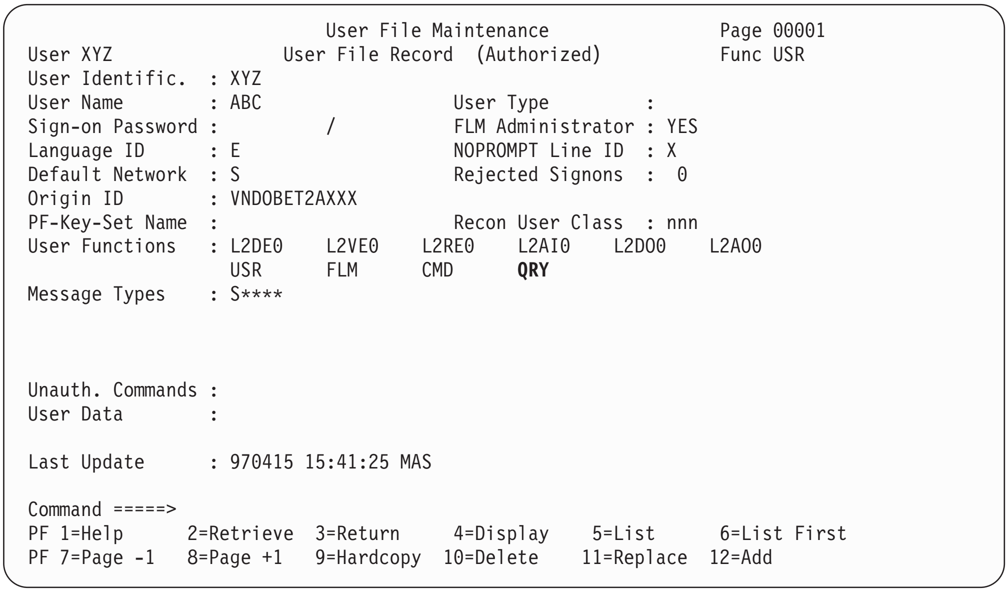
Permissions

In order to use MCB-based queries an end user must have the necessary permissions:

- The query selection function
- A Traffic Reconciliation user class

This is done by updating the user record in the MERVA ESA user file. The query selection function (QRY) must be added to the list of user functions, and a class entered in the **Recon User Class**. The user class must be a number between 1 and 254 excluding, for technical reasons, 64.

The end user is then able to access and run all queries defined in the query table that list this user class in the **CLASS** parameter.



```
                            User File Maintenance                Page 00001
User XYZ                User File Record  (Authorized)           Func USR
User Identific.  : XYZ
User Name        : ABC                    User Type         :
Sign-on Password :        /               FLM Administrator : YES
Language ID      : E                      NOPROMPT Line ID  : X
Default Network  : S                      Rejected Signons  :  0
Origin ID        : VNDOBET2AXXX
PF-Key-Set Name  :                        Recon User Class  : nnn
User Functions   : L2DE0    L2VE0    L2RE0    L2AI0    L2DO0    L2AO0
                   USR      FLM      CMD      QRY
Message Types    : S****



Unauth. Commands :
User Data        :

Last Update      : 970415 15:41:25 MAS

Command =====>
PF 1=Help      2=Retrieve  3=Return     4=Display    5=List      6=List First
PF 7=Page -1   8=Page +1   9=Hardcopy  10=Delete    11=Replace  12=Add
```

Figure 17. Record in the User File

Queue Keys

Traffic Reconciliation provides a separation routine, IMRMS031, number 4031, to derive queue keys from the Traffic Reconciliation system fields, see “Routing of Queries and Results” on page 81. The appropriate TOF fields are defined in the MERVA field definition table:

```
IMRKEY01 DSLLSUBF FIELD=NLEXIT,LENGTH=24,SEPR=4031
IMRKEY02 DSLLSUBF FIELD=NLEXIT,LENGTH=24,SEPR=4031
IMRKEY03 DSLLSUBF FIELD=NLEXIT,LENGTH=24,SEPR=4031
```

The structure of the keys can differ depending on the type of message.

IMRKEY01

IMRKEY01 is defined as follows:

```
  Queries and Lists:                Retrievals:
  Query identification  10          Query identification  10
  Separator              1          Separator              1
  Return code            2          Application/APDU       3
  Separator              1          Separator              1
  Requesting user        8          Correspondent          8
  Blank                  2          Blank                  1
```

The query identification is composed of:

- The letter Q
- The current day (DD)
- The current hour and minute (HHMM)
- The last 3 digits of the MERVA UMR sequence number

IMRKEY02

The structure of field IMRKEY02 is:

```
Queries and Lists:                 Retrievals:
DESCR= from MTT       24           SWIFT message ID       4
                                   Separator              1
                                   MUR or 1.TRN          16
                                   Blank                  3
```

IMRKEY03

The structure of field IMRKEY03 is the same for queries, lists, and responses:

```
Requesting user ID     8
Separator              1
Return code            2
Name of query          8
Separator              1
Processing time HHMM   4
```

To use these keys, they must be specified in the required function definition. For example:

```
DSLFNT NAME=QRYRTV0,QUEUE=YES,SPCMND=(DEL),MSGSEL=LIST,
      KEY1=(IMRKEY01,24),KEY2=(IMRKEY02,24),PROT=YES,
      DESCR='Output Queue for Query Retrievals'
```

The headings in the Queue Key List panel should accord with the key specifications. For queue lists in functions QRYRSP0, QRYRTV0, and QRYLST0 the standard MERVA MCB for queue key list display (DSL0QLI) displays a special heading using these reconciliation keys. You can change DSL0QLI to modify the way these headings are displayed or to display the headings in other functions in the same way.

PF Keys

Function key settings for the query selection function (QRY) are defined in the copybook IMRMPF00. You can change these settings. Note that, to activate the settings, the copybook must be copied by hand into the MERVA ESA function key table (DSLMPF00), as IMRMPF00 is not included by the COPY statement.

Enabling the DELALL Command

The **DELALL** command deletes multiple messages from a queue depending on a key pattern. Messages are deleted by the SQL processor, IMRSQLP, in the background.

To enable the **DELALL** command for a MERVA queue, add the special command **DEL** to the queue specification in the MERVA function table. For example:

```
DSLFNT NAME=QRYRTV0,QUEUE=YES,SPCMND=(DEL),MSGSEL=LIST,
      KEY1=(IMRKEY01,24),KEY2=(IMRKEY02,24),PROT=YES,
      DESCR='Output Queue for Query Retrievals'
```

Traffic Reconciliation also provides the **CLEAR** command to delete all messages produced by the SQL processor (queries, responses, lists, and retrievals) for a particular user from a MERVA queue.

As the **CLEAR** command only deletes messages belonging to the user entering the command, the command is not restricted.

A Complete Example

Traffic Reconciliation includes an additional MCB-based query (QNAK) in source file form in the distributed materials. It is intended to show what you might need to do when defining a new query, and how MERVA MFS exits can be used to solve some common problems.

Refer to the sources as you read this section.

The query is not installed or integrated in any way with other MCB-based queries. You can install it in your system as an exercise.

The Requirement

You want to be able to display a list of FIN input messages that have received a NAK from S.W.I.F.T.. The list should contain the following information:

- When the message was submitted to S.W.I.F.T.
- Message type
- Destination
- TRN
- Session number and sequence number
- NAK code

The newest message must be displayed first.

You want to be able to select messages for the list by any of the following values:

- Message type
- Destination, allowing wildcard characters
- An ISN range
- A date, time range, MMDDHH – MMDDHH
- Number of messages displayed

In addition, these values should be initialized to default values to simplify use of the query:

Message type	100
ISN range	0 – 999999
Date, time range	in the last hour
No. of messages	20

The Solution

First, a query MCB is required (refer to source module IMRU1QNK). The query uses a number of TOF fields that must be defined to MERVA (IMRU1FDT). Messages are not to be retrieved, just a normal SQL result table is required, so a list MCB is also needed to display the result table (IMRU1LNK). These two MCBs must be defined in the MERVA message type table (copybook IMRU1MTT).

The query must be defined to Traffic Reconciliation in the query table (IMRU1QTT). Note that existing routing is used, that is, queue IMRSQLO and its associated routing table IMRSQLRT.

Some of the TOF fields specify MERVA MFS exits. The following exits are defined:

- A default setting exit (IMRMD301) to set the query parameter defaults.

- A checking exit (IMRMC311) to validate the date parameter. Normally you would use a standard MERVA checking exit for a date field, but this date field has a composite format not supported by the standard exits so a specific checking exit has to be written.
- A separation exit to convert the MMDDHH format to the DB2 timestamp format required by a PKEY column (IMRMS331).
- Another separation exit to provide an indication of whether wildcard characters are specified in the destination parameter (IMRMS332). Field QNKQDTYP is then used to control whether the SQL operator **=** or **LIKE** is specified in IMRU1QNK.

There are a number of standard MERVA ESA exits that you might be able to use when creating your own queries. See the section on calling MFS data manipulation programs and exits in the *MERVA for ESA V4 System Programming Guide*.

MFS exits must be defined to MERVA in the MPT (MFS program table). The necessary definitions are in IMRU1MPT.

The checking exit causes MFS to issue an error message if date validation fails by setting the MFS reason code to 500. This requires an error message with number IMR3500 to be defined in the MERVA operator message definitions (IMRU1MSG).

Installation

To install the query you must:

1. Assemble and link-edit the MCBs IMRU1QNK and IMRU1LNK (AMODE=31,RMODE=ANY).
2. Assemble and link-edit the MFS exit programs IMRMD301, IMRMC311, IMRMS331, and IMRMS332 (AMODE=31,RMODE=ANY).
3. Regenerate, that is assemble and link, the MERVA field definition table DSLFDTT including copy book IMRU1FDT.
4. Regenerate the MERVA message type table DSLMTTT including copy book IMRU1MTT.
5. Reassemble the MERVA MFS program table DSLMPTT including copy book IMRU1MPT. The table must be linked into the MERVA MFS program DSLMMFS.
6. Regenerate the MERVA operator message table, DSLMSGT, including copy book IMRU1MSG.
7. Regenerate the query table IMRQRYT including copy book IMRU1QTT.
8. If running in a CICS environment, the exits and MCBs must be defined to CICS so that they can be dynamically loaded (IMRU1CSD).

After installation MERVA ESA users with a reconciliation user class of 1, 3, or 4 will find a new Query (QNAK) in the query list when they next select the QRY function.

Financial Message Capture

Financial Message Capture is a Traffic Reconciliation transaction that can be associated with a MERVA ESA queue to add any S.W.I.F.T. messages routed to that queue to the Traffic Reconciliation S.W.I.F.T. monitor DB2 tables. It allows S.W.I.F.T. messages that are *not* processed by MERVA ESA SWIFT Link to be recorded in the SWIFT Link tables.

Note: As a consequence, the SWIFT Link tables no longer reflect MERVA SWIFT Link traffic, but rather S.W.I.F.T. traffic more generally.

Customizing Financial Message Capture

The Financial Message Capture program (IMRSWFF) handles messages in the same way as the S.W.I.F.T. monitor. That is, it uses the S.W.I.F.T. monitor specifications in IMRPRM and the table filter. You cannot customize Financial Message Capture differently from the S.W.I.F.T. monitor.

Activating Financial Message Capture

Financial Message Capture is activated by associating its transaction name, by default IMRF, with a MERVA queue. To add Financial Message Capture to an existing messaging application, you could define a new queue for this purpose and modify your message routing scheme to route messages also to this queue.

For example, to add Financial Message Capture to an existing FMT/ESA application, you might define two queues: one for succesfully sent messages (for example IMRFMCI), and one for successfully received messages (for example IMRFMCO). Using the sample FMT/ESA routing table EKARTSIM as an example, you could change the routing logic to route all successfully transferred and acknowledged messages to IMRFMCI instead of to EKASWACK. Then, in the function definition of IMRFMCI, you would specify **NEXT=EKASWACK** to route these messages to queue EKASWACK after the messages have been captured by Traffic Reconciliation.

Similarly, received messages would be routed to IMRFMCO instead of EKASWSDO, and IMRFMCO would route messages on to EKASWSDO after processing. FMT/ESA is described in the *MERVA for ESA V4 Customization Guide*.

Differences from SWIFT Link Monitor

Only those messages that are routed through the transaction's queue are captured. If acknowledged messages are routed to the queue, the message is captured, but not the acknowledgment message, unless the acknowledgment is also routed to the queue.

A row containing the queue name associated with the transactions is written to:

- For input messages, the SRC table
- For output messages, the TAR table

Supporting Several MERVA Installations

Events from more than one MERVA ESA system can be inserted into a single Traffic Reconciliation database. If all events are being inserted synchronously into the DB2 tables, no particular measures need be taken.

However, if events are being written to intermediate flip-flop data sets, each MERVA ESA must:

- Have a unique value in the DSLPRM **NAME** parameter
- Use separate VSAM flip-flop data sets
- Run the insertion transaction or program separately (IMRINSP)

Further, if the DSLPRM **NAME** parameter of a MERVA ESA system must at any time be changed:

- All events in the corresponding flip-flop must first be processed by the insertion process. This can be done by running the batch insertion program.
- The VSAM flip-flop must be deleted and re-allocated.
- The row for this MERVA in table IMRCTL should be deleted. The column LSTMRV contains the old name of the renamed MERVA.

Suppressing and Activating Operator Messages

In order to suppress specific operator messages, that is, to keep them from being issued, insert a hyphen (-) before the first character of the message text in the copy book IMRMSGTC.

For example, to suppress the message:

```
IMR077I DSLMSG 'Insertion starts from @0 after RBA = @4'
```

insert a minus sign before the first character of the message text:

```
IMR077I DSLMSG '-Insertion starts from @0 after RBA = @4'
```

Some messages are supressed by default. To activate such messages, remove the hyphen from before the first character of the message text in the copy book IMRMSGTC

Chapter 9. Deleting Traffic Reconciliation Data

Traffic Reconciliation provides a batch utility (IMRRDU) that you can use to delete old data from Traffic Reconciliation DB2 tables. The utility deletes database records inserted into the tables of the Traffic Reconciliation monitor you specify on or before the date you specify. You can specify that it save the deleted records to a VSAM data set. Each database record is comprised of one row of the root table (usually an HDR table), plus all the rows subordinate to it.

Rows for a database record are deleted "bottom up", that is those in the subordinate tables first, so that referential integrity is neither required nor used. There is no advantage in specifying referential integrity for Traffic Reconciliation tables. Subordinate rows with no parent are not deleted. Such "orphan" rows can occur only if the tables have been manipulated externally.

Here is an example of the JCL required to run the delete utility:

```
//DELETE   EXEC PGM=IMRRDU,PARM='MQI,DAYS=100'
//STEPLIB  DD   DSN=recon.SIMRLODB,DISP=SHR
//         DD   DSN=merva.SDSLLODB,DISP=SHR
//         DD   DSN=DB2.LOADLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
//IMROUT   DD   DSN=hlq.IMROUT,DISP=(NEW,CATLG),
//              RECORG=ES,          VSAM ESDS
//              LRECL=32752,
//              SPACE=(3500,(1000,100))  (AVG-RECL,(PRIM RECS,SEC RECS))
```

The program can return the following values to the operating system:

0 The program terminated normally.

8 The program terminated following an unexpected error. An operator message will have been issued describing the error.

12 The program terminated following an internal error.

16 There was insufficient storage to run the program.

Parameters of the Delete Utility

The delete utility is controlled by the PARM parameter in the JCL EXEC statement:

```
   PARM='monitor,{DATE=yyyymmdd}[,COMMIT=nnnn]'
                 {DAYS=nnnn    }
```

The parameters can be given in any order.

monitor The name of the Traffic Reconciliation monitor from whose tables rows are to be deleted (one of: SWIFT, TELEX, MLINK, MQI, QUEUE, JOURNAL).

DATE The delete date in the form *yyyymmdd*. All database records inserted on or before this date are deleted from the specified monitor's tables. The insertion date is the PKEY (not the EKEY) column of the root table.

DAYS Instead of specifying an explicit date, you can specify a number of days (0 to 9999). All database records with an insertion date (root table PKEY) of:

```
todays_date - days
```

or earlier are deleted.

COMMIT A number from 1 to 9999 that specifies the number of database records to be deleted before a DB2 commit is performed.

Writing Deleted Records to an Output Data Set

If a data set with DDname **IMROUT** is specified in the JCL, each deleted database record is written to it in the form of a flip-flop record.

Product-Sensitive Programming Interface

The format of flip-flop records is defined in the copybook IMRFILE. Note that these records are not necessarily identical to the records created by the Traffic Reconciliation extraction process. Data from field tables for queue, MERVA Link, Telex Link, and MQI events are not included in the event records written to IMROUT. Because S.W.I.F.T. fields are extracted from the documentation string, not from a separate field extraction string, S.W.I.F.T. events in IMROUT are complete. Also, the MSGTRACE field for queue events, if recorded, is included in queue event records in IMROUT.

End of Product-Sensitive Programming Interface

The data set must be a VSAM entry-sequenced data set (ESDS) with a maximum record size of 32752. The following JCL creates a new data set of the required form:

```
//IMROUT   DD   DSN=hlq.IMROUT,DISP=(NEW,CATLG),
//              RECORG=ES,         VSAM ESDS
//              LRECL=32752,
//              SPACE=(3500,(1000,100)) (AVG-RECL,(PRIM RECS,SEC RECS))
```

Alternatively, you can specify an existing data set, as shown below, and IMRRDU will append event records to it:

```
//IMROUT   DD   DSN=hlq.IMROUT,DISP=OLD
```

Suppressing the Writing of Deleted Records to an Output Data Set

To supress the writing of deleted records to **IMROUT**, specify it as a DUMMY data set. You cannot simply omit the DD statement, as this will result in an error message. Specify either:

```
//IMROUT   DD   DUMMY,AMP=AMORG

or

//IMROUT   DD   DSN=NULLFILE
```

Sorting Records in the Output Data Set

IMRRDU writes records to IMROUT in ascending sequence of the root table PKEY (that is, the insertion timestamp). However, if you are deleting data for several monitors, and each IMRRDU execution adds event records to a single IMROUT data set, this sequence will not be preserved throughout the data set. You will need to sort the data set records by date to restore the proper event sequence.

An alternative to extending a single IMROUT data set is to create a separate data set for each monitor, then to use DFSORT in a final job step to merge them into a single data set in event sequence.

For more information on DFSORT, refer to the *DFSORT Application Programming Guide*.

Removing Duplicate Records from the Output Data Set

There is no way to coordinate DB2 commits with VSAM. So, to ensure records cannot be lost, IMRRDU performs a VSAM temporary close of IMROUT before each DB2 commit. Consequently, if IMRRDU fails and is restarted, some events might be written a second time to IMROUT. You can use DFSORT to remove such duplicate records in the following way:

```
//SORT  EXEC PGM=ICEMAN
//SYSOUT   DD  SYSOUT=*
//SORTIN   DD  DSN=hlq.IMROUT,DISP=OLD
//SORTOUT  DD  DSN=hlq.IMROUT.SORTED,DISP=(NEW,CATLG,DELETE),
//             RECORG=ES,          VSAM ESDS
//             LRECL=32752,
//             SPACE=(3500,(1000,100))
//SYSIN    DD *
  SORT FIELDS=(5,36,CH,A),DYNALLOC
  RECORD TYPE=V
  SUM FIELDS=NONE
/*
```

For more information on how to use DFSORT to remove duplicate records, refer to the *DFSORT Application Programming Guide*.

Converting the Output Data Set to QSAM

You can use DFSORT to convert the VSAM IMROUT to a QSAM data set if you prefer to archive using QSAM. For more information on how to do this, refer to the *DFSORT Application Programming Guide*.

Statistics

At program termination, IMRRDU writes to SYSPRINT the number of rows deleted from each table. It also reports the number of records written to IMROUT, if any.

For a S.W.I.F.T. monitor, the report has the following form:

```
Delete date: 2000-02-09
Rows deleted:
  FINIxxx tables   DOC:         5    HDR:         3    ACK:         2
                   AMN:         2    FLD:         8    F32:         2
                   MER:         0    MIR:         0    MOR:         0
                   REF:         2    SER:         0    SRC:         2
                   TAR:         1    TRL:         7
  FINOxxx tables   DOC:         5    HDR:         3    ACK:         2
                   AMN:         2    FLD:         8    F32:         2
                   MIR:         0    MOR:         0    MUR:         0
                   REF:         2    SEC:         0    SER:         0
                   SES:         1    TAR:         3    TRL:         6
  GPAIxxx tables   DOC:         3    HDR:         3    ACK:         0
                   FLD:         0    MER:         0    MIR:         0
                   MOR:         0    SER:         0    SRC:         0
                   TAR:         3    TRL:         2
  GPAOxxx tables   DOC:         3    HDR:         3    ACK:         0
                   FLD:         0    MIR:         0    SEC:         0
                   SER:         0    SES:         3    TAR:         3
                   TRL:         0
IMROUT dataset not defined
```

For an MLINK monitor, the report has the following form:

```
Delete date: 2000-02-10
Rows deleted:
  MLKIxxx tables   DOC:        15    HDR:        39
                   CTL:       414    FLD:         6
  MLKOxxx tables   DOC:        24    HDR:        32
                   CTL:       851    FLD:       114
Total IMROUT events:         71
```

Part 4. Appendixes

Appendix A. DB2 Table Overview

This section provides an overview of all DB2 tables used by Traffic Reconciliation. **Recl** is the maximum amount of space one row occupies on disk (refer to the description of SYSIBM.SYSTABLES column RECLENGTH in *DB2 for OS/390 SQL Reference*, DB2 Catalog Tables). Tables with a **Recl** of 0 are views.

Table 3. DB2 Table Overview

Table	Recl	Seq	Column	Type	Length	Indexes	Seq
FINIACK	71	1	PKEY	TIMESTMP	10	XFINIACK	1
		2	EKEY	TIMESTMP	10		
		3	DKEY	TIMESTMP	10		
		4	B_BANK	CHAR	4	XFINIACK_BIC	1
		5	B_COUNTRY	CHAR	2		2
		6	B_LOCATION	CHAR	2		3
		7	B_TERMINAL	CHAR	1		4
		8	B_BRANCH	CHAR	3		5
		9	B_SESSION	CHAR	4		
		10	B_SEQUENCE	CHAR	6		
		11	ACK_DATE	CHAR	6		
		12	ACK_TIME	CHAR	4		
		13	F_451	CHAR	1		
FINIACK2	0	1	PKEY	TIMESTMP	10		
		2	EKEY	TIMESTMP	10		
		3	DKEY	TIMESTMP	10		
		4	B_LT	CHAR	12		
		5	B_SESSION	CHAR	4		
		6	B_SEQUENCE	CHAR	6		
		7	ACK_DATE	CHAR	6		
		8	ACK_TIME	CHAR	4		
		9	F_451	CHAR	1		
FINIAMN	51	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XFINIAMN	1
		3	FIELD_ID	CHAR	3		
		4	AMOUNT	DECIMAL	15	XFINIAMN_AMT	1
		5	AMOUNTD	DECIMAL	4		
		6	CURRENCY	CHAR	3	XFINIAMN_CURR	1
		7	DATE	CHAR	6		
FINIDOC	11020	1	PKEY	TIMESTMP	10	XFINIDOC	1
		2	APDU	VARCHAR	11000		
FINIFLD	102	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XFINIFLD	1
		3	FIELD_ID	CHAR	3	XFINIFLD_ID	1
		4	FIELD_DATA	VARCHAR	69		
FINIF32	64	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XFINIF32	1
		3	TRN	CHAR	16		
		4	AMOUNT	DECIMAL	15	XFINIF32_AMT	1
		5	AMOUNTD	DECIMAL	4		
		6	CURRENCY	CHAR	3	XFINIF32_CURR	1
		7	DATE	CHAR	6		

Table 3. DB2 Table Overview (continued)

Table	Recl	Seq	Column	Type	Length	Indexes	Seq
FINIHDR	140	1	PKEY	TIMESTMP	10	XFINIHDR	1
		2	EKEY	TIMESTMP	10	XFINIHDR_EKEY	1
		3	DKEY	TIMESTMP	10		
		4	UMR	CHAR	28	XFINIHDR_UMR	1
		5	APDU_LENGTH	SMALLINT	2		
		6	LINE_NUMBER	SMALLINT	2		
		7	RETRIEVAL	CHAR	1		
		8	MAST_DESTIN	CHAR	8		
		9	MUR_TRN	CHAR	1		
		10	B_APPLID	CHAR	1		
		11	B_APDUID	CHAR	2		
		12	B_BANK	CHAR	4	XFINIHDR_BIC	1
		13	B_COUNTRY	CHAR	2		2
		14	B_LOCATION	CHAR	2		3
		15	B_TERMINAL	CHAR	1		4
		16	B_BRANCH	CHAR	3		5
		17	B_SESSION	CHAR	4		
		18	B_SEQUENCE	CHAR	6		
		19	A_MSGTYPE	CHAR	3		
		20	A_DESTIN	CHAR	8		
		21	A_BRANCH	CHAR	3		
		22	A_PRIORITY	CHAR	1		
		23	A_DELIVERY	CHAR	1		
		24	A_OBSOLES	CHAR	3		
		25	U_MUR	CHAR	16	XFINIHDR_TRN	1
FINIHDR2	0	1	PKEY	TIMESTMP	10		
		2	EKEY	TIMESTMP	10		
		3	DKEY	TIMESTMP	10		
		4	UMR	CHAR	28		
		5	APDU_LENGTH	SMALLINT	2		
		6	LINE_NUMBER	SMALLINT	2		
		7	RETRIEVAL	CHAR	1		
		8	MAST_DESTIN	CHAR	8		
		9	MUR_TRN	CHAR	1		
		10	B_APPLID	CHAR	1		
		11	B_APDUID	CHAR	2		
		12	B_LT	CHAR	12		
		13	B_SESSION	CHAR	4		
		14	B_SEQUENCE	CHAR	6		
		15	A_MSGTYPE	CHAR	3		
		16	A_DESTIN	CHAR	8		
		17	A_BRANCH	CHAR	3		
		18	A_PRIORITY	CHAR	1		
		19	A_DELIVERY	CHAR	1		
		20	A_OBSOLES	CHAR	3		
		21	U_MUR	CHAR	16		
FINIMER	128	1	PKEY	TIMESTMP	10		
		2	UMR	CHAR	28	XFINIMER_UMR	1
		3	ERROR	VARCHAR	80		

Table 3. DB2 Table Overview (continued)

Table	Recl	Seq	Column	Type	Length	Indexes	Seq
FINIMIR	84	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XFINIMIR	1
		3	RELATED_MIR	CHAR	28		
		4	MIR_DATE	CHAR	6		
		5	MIR_BANK	CHAR	4		
		6	MIR_COUNTRY	CHAR	2		
		7	MIR_LOCATION	CHAR	2		
		8	MIR_TERMINAL	CHAR	1		
		9	MIR_BRANCH	CHAR	3		
		10	MIR_SESSION	CHAR	4		
		11	MIR_SEQUENCE	CHAR	6		
FINIMIR2	0	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10		
		3	RELATED_MIR	CHAR	28		
		4	MIR_DATE	CHAR	6		
		5	MIR_LT	CHAR	12		
		6	MIR_SESSION	CHAR	4		
		7	MIR_SEQUENCE	CHAR	6		
FINIMOR	84	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XFINIMOR	1
		3	RELATED_MOR	CHAR	28		
		4	MOR_DATE	CHAR	6		
		5	MOR_BANK	CHAR	4		
		6	MOR_COUNTRY	CHAR	2		
		7	MOR_LOCATION	CHAR	2		
		8	MOR_TERMINAL	CHAR	1		
		9	MOR_BRANCH	CHAR	3		
		10	MOR_SESSION	CHAR	4		
		11	MOR_SEQUENCE	CHAR	6		
FINIMOR2	0	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10		
		3	RELATED_MOR	CHAR	28		
		4	MOR_DATE	CHAR	6		
		5	MOR_LT	CHAR	12		
		6	MOR_SESSION	CHAR	4		
		7	MOR_SEQUENCE	CHAR	6		
FINIREF	47	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XFINIREF	1
		3	FIELD_ID	CHAR	3	XFINIREF_ID	1
		4	REFERENCE	CHAR	16	XFINIREF_DATA	1
FINISER	55	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XFINISER	1
		3	F_401	CHAR	2		
		4	F_405	CHAR	3		
		5	F_421	CHAR	3		
		6	F_431	CHAR	2		
		7	F_432	CHAR	2		
		8	F_441	CHAR	3		
		9	F_442	CHAR	2		
		10	F_443	CHAR	3		
		11	F_451	CHAR	1		
		12	F_461	CHAR	3		
		13	F_503	CHAR	3		

Table 3. DB2 Table Overview (continued)

Table	Recl	Seq	Column	Type	Length	Indexes	Seq
FINISRC	58	1	PKEY	TIMESTMP	10		
		2	UMR	CHAR	28	XFINISRC_UMR	1
		3	QUEUE	CHAR	8		
		4	QSN	INTEGER	4		
FINITAR	58	1	PKEY	TIMESTMP	10		
		2	UMR	CHAR	28	XFINITAR_UMR	1
		3	QUEUE	CHAR	8		
		4	QSN	INTEGER	4		
FINITRL	65	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XFINITRL	1
		3	TRL_ID	CHAR	3		
		4	TRL_TEXT	VARCHAR	32		
FINOACK	71	1	PKEY	TIMESTMP	10	XFINOACK	1
		2	EKEY	TIMESTMP	10		
		3	DKEY	TIMESTMP	10		
		4	B_BANK	CHAR	4	XFINOACK_BIC	1
		5	B_COUNTRY	CHAR	2		2
		6	B_LOCATION	CHAR	2		3
		7	B_TERMINAL	CHAR	1		4
		8	B_BRANCH	CHAR	3		5
		9	B_SESSION	CHAR	4		
		10	B_SEQUENCE	CHAR	6		
		11	ACK_DATE	CHAR	6		
		12	ACK_TIME	CHAR	4		
		13	F_451	CHAR	1		
FINOACK2	0	1	PKEY	TIMESTMP	10		
		2	EKEY	TIMESTMP	10		
		3	DKEY	TIMESTMP	10		
		4	B_LT	CHAR	12		
		5	B_SESSION	CHAR	4		
		6	B_SEQUENCE	CHAR	6		
		7	ACK_DATE	CHAR	6		
		8	ACK_TIME	CHAR	4		
		9	F_451	CHAR	1		
FINOAMN	51	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XFINOAMN	1
		3	FIELD_ID	CHAR	3		
		4	AMOUNT	DECIMAL	15	XFINOAMN_AMT	1
		5	AMOUNTD	DECIMAL	4		
		6	CURRENCY	CHAR	3	XFINOAMN_CURR	1
		7	DATE	CHAR	6		
FINODOC	11020	1	PKEY	TIMESTMP	10	XFINODOC	1
		2	APDU	VARCHAR	11000		
FINOFLD	102	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XFINOFLD	1
		3	FIELD_ID	CHAR	3	XFINOFLD_ID	1
		4	FIELD_DATA	VARCHAR	69		
FINOF32	64	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XFINOF32	1
		3	TRN	CHAR	16		
		4	AMOUNT	DECIMAL	15	XFINOF32_AMT	1
		5	AMOUNTD	DECIMAL	4		
		6	CURRENCY	CHAR	3	XFINOF32_CURR	1
		7	DATE	CHAR	6		

Table 3. DB2 Table Overview (continued)

Table	Recl	Seq	Column	Type	Length	Indexes	Seq
FINOHDR	195	1	PKEY	TIMESTMP	10	XFINOHDR	1
		2	EKEY	TIMESTMP	10	XFINOHDR_EKEY	1
		3	DKEY	TIMESTMP	10		
		4	UMR	CHAR	28	XFINOHDR_UMR	1
		5	APDU_LENGTH	SMALLINT	2		
		6	LINE_NUMBER	SMALLINT	2		
		7	RETRIEVAL	CHAR	1		
		8	MAST_DESTIN	CHAR	8		
		9	MUR_TRN	CHAR	1		
		10	B_APPLID	CHAR	1		
		11	B_APDUID	CHAR	2		
		12	B_BANK	CHAR	4	XFINOHDR_BIC	1
		13	B_COUNTRY	CHAR	2		2
		14	B_LOCATION	CHAR	2		3
		15	B_TERMINAL	CHAR	1		4
		16	B_BRANCH	CHAR	3		5
		17	B_SESSION	CHAR	4		
		18	B_SEQUENCE	CHAR	6		
		19	A_MSGTYPE	CHAR	3		
		20	A_INTIME	CHAR	4		
		21	A_MIR	CHAR	28		
		22	A_MIR_DATE	CHAR	6		
		23	A_MIR_BANK	CHAR	4		
		24	A_MIR_COUNTRY	CHAR	2		
		25	A_MIR_LOCATION	CHAR	2		
		26	A_MIR_TERMINAL	CHAR	1		
		27	A_MIR_BRANCH	CHAR	3		
		28	A_MIR_SESSION	CHAR	4		
		29	A_MIR_SEQUENCE	CHAR	6		
		30	A_OUTDATE	CHAR	6		
		31	A_OUTTIME	CHAR	4		
		32	A_PRIORITY	CHAR	1		
		33	U_MUR	CHAR	16		

Table 3. DB2 Table Overview (continued)

Table	Recl	Seq	Column	Type	Length	Indexes	Seq
FINOHDR2	0	1	PKEY	TIMESTMP	10		
		2	EKEY	TIMESTMP	10		
		3	DKEY	TIMESTMP	10		
		4	UMR	CHAR	28		
		5	APDU_LENGTH	SMALLINT	2		
		6	LINE_NUMBER	SMALLINT	2		
		7	RETRIEVAL	CHAR	1		
		8	MAST_DESTIN	CHAR	8		
		9	MUR_TRN	CHAR	1		
		10	B_APPLID	CHAR	1		
		11	B_APDUID	CHAR	2		
		12	B_LT	CHAR	12		
		13	B_SESSION	CHAR	4		
		14	B_SEQUENCE	CHAR	6		
		15	A_MSGTYPE	CHAR	3		
		16	A_INTIME	CHAR	4		
		17	A_MIR	CHAR	28		
		18	A_MIR_DATE	CHAR	6		
		19	A_MIR_LT	CHAR	12		
		20	A_MIR_SESSION	CHAR	4		
		21	A_MIR_SEQUENCE	CHAR	6		
		22	A_OUTDATE	CHAR	6		
		23	A_OUTTIME	CHAR	4		
		24	A_PRIORITY	CHAR	1		
		25	U_MUR	CHAR	16		
FINOMIR	84	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XFINOMIR	1
		3	RELATED_MIR	CHAR	28		
		4	MIR_DATE	CHAR	6		
		5	MIR_BANK	CHAR	4		
		6	MIR_COUNTRY	CHAR	2		
		7	MIR_LOCATION	CHAR	2		
		8	MIR_TERMINAL	CHAR	1		
		9	MIR_BRANCH	CHAR	3		
		10	MIR_SESSION	CHAR	4		
		11	MIR_SEQUENCE	CHAR	6		
FINOMIR2	0	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10		
		3	RELATED_MIR	CHAR	28		
		4	MIR_DATE	CHAR	6		
		5	MIR_LT	CHAR	12		
		6	MIR_SESSION	CHAR	4		
		7	MIR_SEQUENCE	CHAR	6		
FINOMOR	84	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XFINOMOR	1
		3	RELATED_MOR	CHAR	28		
		4	MOR_DATE	CHAR	6		
		5	MOR_BANK	CHAR	4		
		6	MOR_COUNTRY	CHAR	2		
		7	MOR_LOCATION	CHAR	2		
		8	MOR_TERMINAL	CHAR	1		
		9	MOR_BRANCH	CHAR	3		
		10	MOR_SESSION	CHAR	4		
		11	MOR_SEQUENCE	CHAR	6		

Table 3. DB2 Table Overview (continued)

Table	Recl	Seq	Column	Type	Length	Indexes	Seq
FINOMOR2	0	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10		
		3	RELATED_MOR	CHAR	28		
		4	MOR_DATE	CHAR	6		
		5	MOR_LT	CHAR	12		
		6	MOR_SESSION	CHAR	4		
		7	MOR_SEQUENCE	CHAR	6		
FINOMUR	44	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XFINOMUR	1
		3	RELATED_MUR	CHAR	16		
FINOREF	47	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XFINOREF	1
		3	FIELD_ID	CHAR	3	XFINOREF_ID	1
		4	REFERENCE	CHAR	16	XFINOREF_DATA	1
FINOSEC	36	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XFINOSEC	1
		3	F_202	CHAR	4		
		4	F_203	CHAR	4		
FINOSER	55	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XFINOSER	1
		3	F_401	CHAR	2		
		4	F_405	CHAR	3		
		5	F_421	CHAR	3		
		6	F_431	CHAR	2		
		7	F_432	CHAR	2		
		8	F_441	CHAR	3		
		9	F_442	CHAR	2		
		10	F_443	CHAR	3		
		11	F_451	CHAR	1		
		12	F_461	CHAR	3		
		13	F_503	CHAR	3		
FINOSES	40	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XFINOSES	1
		3	LAST_ISN	CHAR	6		
		4	LAST_OSN	CHAR	6		
FINOTAR	58	1	PKEY	TIMESTMP	10		
		2	UMR	CHAR	28	XFINOTAR_UMR	1
		3	QUEUE	CHAR	8		
		4	QSN	INTEGER	4		
FINOTRL	65	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XFINOTRL	1
		3	TRL_ID	CHAR	3		
		4	TRL_TEXT	VARCHAR	32		

Table 3. DB2 Table Overview (continued)

Table	Recl	Seq	Column	Type	Length	Indexes	Seq
GPAIACK	71	1	PKEY	TIMESTMP	10	XGPAIACK	1
		2	EKEY	TIMESTMP	10		
		3	DKEY	TIMESTMP	10		
		4	B_BANK	CHAR	4	XGPAIACK_BIC	1
		5	B_COUNTRY	CHAR	2		2
		6	B_LOCATION	CHAR	2		3
		7	B_TERMINAL	CHAR	1		4
		8	B_BRANCH	CHAR	3		5
		9	B_SESSION	CHAR	4		
		10	B_SEQUENCE	CHAR	6		
		11	ACK_DATE	CHAR	6		
		12	ACK_TIME	CHAR	4		
		13	F_451	CHAR	1		
GPAIACK2	0	1	PKEY	TIMESTMP	10		
		2	EKEY	TIMESTMP	10		
		3	DKEY	TIMESTMP	10		
		4	B_LT	CHAR	12		
		5	B_SESSION	CHAR	4		
		6	B_SEQUENCE	CHAR	6		
		7	ACK_DATE	CHAR	6		
		8	ACK_TIME	CHAR	4		
		9	F_451	CHAR	1		
GPAIDOC	11020	1	PKEY	TIMESTMP	10	XGPAIDOC	1
		2	APDU	VARCHAR	11000		
GPAIFLD	102	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XGPAIFLD	1
		3	FIELD_ID	CHAR	3	XGPAIFLD_ID	1
		4	FIELD_DATA	VARCHAR	69		
GPAIHDR	118	1	PKEY	TIMESTMP	10	XGPAIHDR	1
		2	EKEY	TIMESTMP	10	XGPAIHDR_EKEY	1
		3	DKEY	TIMESTMP	10		
		4	UMR	CHAR	28	XGPAIHDR_UMR	1
		5	APDU_LENGTH	SMALLINT	2		
		6	LINE_NUMBER	SMALLINT	2		
		7	RETRIEVAL	CHAR	1		
		8	MAST_DESTIN	CHAR	8		
		9	B_APPLID	CHAR	1		
		10	B_APDUID	CHAR	2		
		11	B_BANK	CHAR	4	XGPAIHDR_BIC	1
		12	B_COUNTRY	CHAR	2		2
		13	B_LOCATION	CHAR	2		3
		14	B_TERMINAL	CHAR	1		4
		15	B_BRANCH	CHAR	3		5
		16	B_SESSION	CHAR	4		
		17	B_SEQUENCE	CHAR	6		
		18	A_MSGTYPE	CHAR	3		
		19	A_DESTIN	CHAR	8		
		20	A_BRANCH	CHAR	3		

Table 3. DB2 Table Overview (continued)

Table	Recl	Seq	Column	Type	Length	Indexes	Seq
GPAIHDR2	0	1	PKEY	TIMESTMP	10		
		2	EKEY	TIMESTMP	10		
		3	DKEY	TIMESTMP	10		
		4	UMR	CHAR	28		
		5	APDU_LENGTH	SMALLINT	2		
		6	LINE_NUMBER	SMALLINT	2		
		7	RETRIEVAL	CHAR	1		
		8	MAST_DESTIN	CHAR	8		
		9	B_APPLID	CHAR	1		
		10	B_APDUID	CHAR	2		
		11	B_LT	CHAR	12		
		12	B_SESSION	CHAR	4		
		13	B_SEQUENCE	CHAR	6		
		14	A_MSGTYPE	CHAR	3		
		15	A_DESTIN	CHAR	8		
		16	A_BRANCH	CHAR	3		
GPAIMER	128	1	PKEY	TIMESTMP	10		
		2	UMR	CHAR	28	XGPAIMER_UMR	1
		3	ERROR	VARCHAR	80		
GPAIMIR	84	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XGPAIMIR	1
		3	RELATED_MIR	CHAR	28		
		4	MIR_DATE	CHAR	6		
		5	MIR_BANK	CHAR	4		
		6	MIR_COUNTRY	CHAR	2		
		7	MIR_LOCATION	CHAR	2		
		8	MIR_TERMINAL	CHAR	1		
		9	MIR_BRANCH	CHAR	3		
		10	MIR_SESSION	CHAR	4		
		11	MIR_SEQUENCE	CHAR	6		
GPAIMIR2	0	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10		
		3	RELATED_MIR	CHAR	28		
		4	MIR_DATE	CHAR	6		
		5	MIR_LT	CHAR	12		
		6	MIR_SESSION	CHAR	4		
		7	MIR_SEQUENCE	CHAR	6		
GPAIMOR	84	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XGPAIMOR	1
		3	RELATED_MOR	CHAR	28		
		4	MOR_DATE	CHAR	6		
		5	MOR_BANK	CHAR	4		
		6	MOR_COUNTRY	CHAR	2		
		7	MOR_LOCATION	CHAR	2		
		8	MOR_TERMINAL	CHAR	1		
		9	MOR_BRANCH	CHAR	3		
		10	MOR_SESSION	CHAR	4		
		11	MOR_SEQUENCE	CHAR	6		
GPAIMOR2	0	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10		
		3	RELATED_MOR	CHAR	28		
		4	MOR_DATE	CHAR	6		
		5	MOR_LT	CHAR	12		
		6	MOR_SESSION	CHAR	4		
		7	MOR_SEQUENCE	CHAR	6		

Table 3. DB2 Table Overview (continued)

Table	Recl	Seq	Column	Type	Length	Indexes	Seq
GPAISER	55	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XGPAISER	1
		3	F_401	CHAR	2		
		4	F_405	CHAR	3		
		5	F_421	CHAR	3		
		6	F_431	CHAR	2		
		7	F_432	CHAR	2		
		8	F_441	CHAR	3		
		9	F_442	CHAR	2		
		10	F_443	CHAR	3		
		11	F_451	CHAR	1		
		12	F_461	CHAR	3		
		13	F_503	CHAR	3		
GPAISRC	58	1	PKEY	TIMESTMP	10		
		2	UMR	CHAR	28	XGPAISRC_UMR	1
		3	QUEUE	CHAR	8		
		4	QSN	INTEGER	4		
GPAITAR	58	1	PKEY	TIMESTMP	10		
		2	UMR	CHAR	28	XGPAITAR_UMR	1
		3	QUEUE	CHAR	8		
		4	QSN	INTEGER	4		
GPAITRL	65	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XGPAITRL	1
		3	TRL_ID	CHAR	3		
		4	TRL_TEXT	VARCHAR	32		
GPAOACK	71	1	PKEY	TIMESTMP	10	XGPAOACK	1
		2	EKEY	TIMESTMP	10		
		3	DKEY	TIMESTMP	10		
		4	B_BANK	CHAR	4	XGPAOACK_BIC	1
		5	B_COUNTRY	CHAR	2		2
		6	B_LOCATION	CHAR	2		3
		7	B_TERMINAL	CHAR	1		4
		8	B_BRANCH	CHAR	3		5
		9	B_SESSION	CHAR	4		
		10	B_SEQUENCE	CHAR	6		
		11	ACK_DATE	CHAR	6		
		12	ACK_TIME	CHAR	4		
		13	F_451	CHAR	1		
GPAOACK2	0	1	PKEY	TIMESTMP	10		
		2	EKEY	TIMESTMP	10		
		3	DKEY	TIMESTMP	10		
		4	B_LT	CHAR	12		
		5	B_SESSION	CHAR	4		
		6	B_SEQUENCE	CHAR	6		
		7	ACK_DATE	CHAR	6		
		8	ACK_TIME	CHAR	4		
		9	F_451	CHAR	1		
GPAODOC	11020	1	PKEY	TIMESTMP	10	XGPAODOC	1
		2	APDU	VARCHAR	11000		
GPAOFLD	102	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XGPAOFLD	1
		3	FIELD_ID	CHAR	3	XGPAOFLD_ID	1
		4	FIELD_DATA	VARCHAR	69		

Table 3. DB2 Table Overview (continued)

Table	Recl	Seq	Column	Type	Length	Indexes	Seq
GPAOHDR	177	1	PKEY	TIMESTMP	10	XGPAOHDR	1
		2	EKEY	TIMESTMP	10	XGPAOHDR_EKEY	1
		3	DKEY	TIMESTMP	10		
		4	UMR	CHAR	28	XGPAOHDR_UMR	1
		5	APDU_LENGTH	SMALLINT	2		
		6	LINE_NUMBER	SMALLINT	2		
		7	RETRIEVAL	CHAR	1		
		8	MAST_DESTIN	CHAR	8		
		9	B_APPLID	CHAR	1		
		10	B_APDUID	CHAR	2		
		11	B_BANK	CHAR	4	XGPAOHDR_BIC	1
		12	B_COUNTRY	CHAR	2		2
		13	B_LOCATION	CHAR	2		3
		14	B_TERMINAL	CHAR	1		4
		15	B_BRANCH	CHAR	3		5
		16	B_SESSION	CHAR	4		
		17	B_SEQUENCE	CHAR	6		
		18	A_MSGTYPE	CHAR	3		
		19	A_INTIME	CHAR	4		
		20	A_MIR	CHAR	28		
		21	A_MIR_DATE	CHAR	6		
		22	A_MIR_BANK	CHAR	4		
		23	A_MIR_COUNTRY	CHAR	2		
		24	A_MIR_LOCATION	CHAR	2		
		25	A_MIR_TERMINAL	CHAR	1		
		26	A_MIR_BRANCH	CHAR	3		
		27	A_MIR_SESSION	CHAR	4		
		28	A_MIR_SEQUENCE	CHAR	6		
		29	A_OUTDATE	CHAR	6		
		30	A_OUTTIME	CHAR	4		
GPAOHDR2	0	1	PKEY	TIMESTMP	10		
		2	EKEY	TIMESTMP	10		
		3	DKEY	TIMESTMP	10		
		4	UMR	CHAR	28		
		5	APDU_LENGTH	SMALLINT	2		
		6	LINE_NUMBER	SMALLINT	2		
		7	RETRIEVAL	CHAR	1		
		8	MAST_DESTIN	CHAR	8		
		9	B_APPLID	CHAR	1		
		10	B_APDUID	CHAR	2		
		11	B_LT	CHAR	12		
		12	B_SESSION	CHAR	4		
		13	B_SEQUENCE	CHAR	6		
		14	A_MSGTYPE	CHAR	3		
		15	A_INTIME	CHAR	4		
		16	A_MIR	CHAR	28		
		17	A_MIR_DATE	CHAR	6		
		18	A_MIR_LT	CHAR	12		
		19	A_MIR_SESSION	CHAR	4		
		20	A_MIR_SEQUENCE	CHAR	6		
		21	A_OUTDATE	CHAR	6		
		22	A_OUTTIME	CHAR	4		

Table 3. DB2 Table Overview (continued)

Table	Recl	Seq	Column	Type	Length	Indexes	Seq
GPAOMIR	84	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XGPAOMIR	1
		3	RELATED_MIR	CHAR	28		
		4	MIR_DATE	CHAR	6		
		5	MIR_BANK	CHAR	4		
		6	MIR_COUNTRY	CHAR	2		
		7	MIR_LOCATION	CHAR	2		
		8	MIR_TERMINAL	CHAR	1		
		9	MIR_BRANCH	CHAR	3		
		10	MIR_SESSION	CHAR	4		
		11	MIR_SEQUENCE	CHAR	6		
GPAOMIR2	0	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10		
		3	RELATED_MIR	CHAR	28		
		4	MIR_DATE	CHAR	6		
		5	MIR_LT	CHAR	12		
		6	MIR_SESSION	CHAR	4		
		7	MIR_SEQUENCE	CHAR	6		
GPAOSEC	36	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XGPAOSEC	1
		3	F_202	CHAR	4		
		4	F_203	CHAR	4		
GPAOSER	55	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XGPAOSER	1
		3	F_401	CHAR	2		
		4	F_405	CHAR	3		
		5	F_421	CHAR	3		
		6	F_431	CHAR	2		
		7	F_432	CHAR	2		
		8	F_441	CHAR	3		
		9	F_442	CHAR	2		
		10	F_443	CHAR	3		
		11	F_451	CHAR	1		
		12	F_461	CHAR	3		
		13	F_503	CHAR	3		
GPAOSES	40	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XGPAOSES	1
		3	LAST_ISN	CHAR	6		
		4	LAST_OSN	CHAR	6		
GPAOTAR	58	1	PKEY	TIMESTMP	10		
		2	UMR	CHAR	28	XGPAOTAR_UMR	1
		3	QUEUE	CHAR	8		
		4	QSN	INTEGER	4		
GPAOTRL	65	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XGPAOTRL	1
		3	TRL_ID	CHAR	3		
		4	TRL_TEXT	VARCHAR	32		
IMRCTL	40	1	LSTMRV	CHAR	8	XIMRCTL	1
		2	LSTRBA	INTEGER	4		
		3	LSTFID	CHAR	20		

Table 3. DB2 Table Overview (continued)

Table	Recl	Seq	Column	Type	Length	Indexes	Seq
JRNPUT	16070	1	PKEY	TIMESTMP	10	XJRNPUT	1
		2	EKEY	TIMESTMP	10	XJRNPUT_EKEY	1
		3	JRN_ID	CHAR	1	XJRNPUT_ID	1
		4	JRN_DATE	CHAR	8		
		5	JRN_TIME	CHAR	6		
		6	JRN_USER	CHAR	25		
		7	JRN_DATA	VARCHAR	16000		
MLKICTL	294	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XMLKICTL	1
		3	CTL_ID	CHAR	8		
		4	CTL_DATA	VARCHAR	256		
MLKIDOC	11030	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XMLKIDOC	1
		3	APDU	VARCHAR	11000		
MLKIFLD	294	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XMLKIFLD	1
		3	FIELD_ID	CHAR	8		
		4	FIELD_DATA	VARCHAR	256		
MLKIHDR	92	1	PKEY	TIMESTMP	10	XMLKIHDR	1
		2	EKEY	TIMESTMP	10		
		3	ASP_NAME	CHAR	8		
		4	EVENT	CHAR	1		
		5	MSG_CATEGORY	CHAR	1		
		6	UMR	CHAR	28		
		7	MSG_TYPE	CHAR	8		
		8	MSG_STATUS	CHAR	2		
		9	MSG_IAM	CHAR	16		
MLKOCTL	294	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XMLKOCTL	1
		3	CTL_ID	CHAR	8		
		4	CTL_DATA	VARCHAR	256		
MLKODOC	11030	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XMLKODOC	1
		3	APDU	VARCHAR	11000		
MLKOFLD	294	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XMLKOFLD	1
		3	FIELD_ID	CHAR	8		
		4	FIELD_DATA	VARCHAR	256		
MLKOHDR	92	1	PKEY	TIMESTMP	10	XMLKOHDR	1
		2	EKEY	TIMESTMP	10		
		3	ASP_NAME	CHAR	8		
		4	EVENT	CHAR	1		
		5	MSG_CATEGORY	CHAR	1		
		6	UMR	CHAR	28		
		7	MSG_TYPE	CHAR	8		
		8	MSG_STATUS	CHAR	2		
		9	MSG_IAM	CHAR	16		
MQIIDOC	11030	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XMQIIDOC	1
		3	APDU	VARCHAR	11000		

Table 3. DB2 Table Overview (continued)

Table	Recl	Seq	Column	Type	Length	Indexes	Seq
MQIIDOF	294	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XMQIIDOF	1
		3	FIELD_ID	CHAR	8	XMQIIDOF_FIELD	1
		4	FIELD_DATA	VARCHAR	256		
MQIIFLD	294	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XMQIIFLD	1
		3	FIELD_ID	CHAR	8	XMQIIFLD_FIELD	1
		4	FIELD_DATA	VARCHAR	256		
MQIIHDR	512	1	PKEY	TIMESTMP	10	XMQIIHDR	1
		2	EKEY	TIMESTMP	10	XMQIIHDR_EKEY	1
		3	MTYPE	CHAR	8		
		4	UMR	CHAR	28	XMQIIHDR_UMR	1
		5	PROCESS	CHAR	8	XMQIIHDR_PROCESS	1
		6	EVENT	CHAR	8		
		7	DOC_FIELDS	INTEGER	4		
		8	REPORT	INTEGER	4		
		9	EXPIRY	INTEGER	4		
		10	FEEDBACK	INTEGER	4		
		11	ENCODING	INTEGER	4		
		12	CCSID	INTEGER	4		
		13	FORMAT	CHAR	8		
		14	PRIORITY	INTEGER	4		
		15	PERSISTENCE	INTEGER	4		
		16	MSGID	CHAR	24		
		17	CORRELID	CHAR	24		
		18	BACKOUT	INTEGER	4		
		19	USERID	CHAR	12		
		20	ACCOUNTING	CHAR	32		
		21	APPLIDENT	CHAR	32		
		22	PUTAPPLTYPE	INTEGER	4		
		23	PUTAPPLNAME	CHAR	28		
		24	PUTDATE	CHAR	8		
		25	PUTTIME	CHAR	8		
		26	APPLORIGIN	CHAR	4		
		27	OPTIONSID	CHAR	4		
		28	OPTIONS	INTEGER	4		
		29	WAITINTERVAL	INTEGER	4		
		30	REPLYTOQ	VARCHAR	48		
		31	REPLYTOQMGR	VARCHAR	48		
		32	RESOLVEDQ	VARCHAR	48		
		33	RESOLVEDQMGR	VARCHAR	48		

Table 3. DB2 Table Overview (continued)

Table	Recl	Seq	Column	Type	Length	Indexes	Seq
MQIIHDR_TEMP	512	1	PKEY	TIMESTMP	10		
		2	EKEY	TIMESTMP	10		
		3	MTYPE	CHAR	8		
		4	UMR	CHAR	28		
		5	PROCESS	CHAR	8	XMQIIHDRT_PROCESS	1
		6	EVENT	CHAR	8		
		7	DOC_FIELDS	INTEGER	4		
		8	REPORT	INTEGER	4		
		9	EXPIRY	INTEGER	4		
		10	FEEDBACK	INTEGER	4		
		11	ENCODING	INTEGER	4		
		12	CCSID	INTEGER	4		
		13	FORMAT	CHAR	8		
		14	PRIORITY	INTEGER	4		
		15	PERSISTENCE	INTEGER	4		
		16	MSGID	CHAR	24		
		17	CORRELID	CHAR	24		
		18	BACKOUT	INTEGER	4		
		19	USERID	CHAR	12		
		20	ACCOUNTING	CHAR	32		
		21	APPLIDENT	CHAR	32		
		22	PUTAPPLTYPE	INTEGER	4		
		23	PUTAPPLNAME	CHAR	28		
		24	PUTDATE	CHAR	8		
		25	PUTTIME	CHAR	8		
		26	APPLORIGIN	CHAR	4		
		27	OPTIONSID	CHAR	4		
		28	OPTIONS	INTEGER	4		
		29	WAITINTERVAL	INTEGER	4		
		30	REPLYTOQ	VARCHAR	48		
		31	REPLYTOQMGR	VARCHAR	48		
		32	RESOLVEDQ	VARCHAR	48		
		33	RESOLVEDQMGR	VARCHAR	48		
MQIODOC	11030	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XMQIODOC	1
		3	APDU	VARCHAR	11000		
MQIODOF	294	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XMQIODOF	1
		3	FIELD_ID	CHAR	8	XMQIODOF_FIELD	1
		4	FIELD_DATA	VARCHAR	256		
MQIOFLD	294	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XMQIOFLD	1
		3	FIELD_ID	CHAR	8	XMQIOFLD_FIELD	1
		4	FIELD_DATA	VARCHAR	256		

Table 3. DB2 Table Overview (continued)

Table	Recl	Seq	Column	Type	Length	Indexes	Seq
MQIOHDR	512	1	PKEY	TIMESTMP	10	XMQIOHDR	1
		2	EKEY	TIMESTMP	10	XMQIOHDR_EKEY	1
		3	MTYPE	CHAR	8		
		4	UMR	CHAR	28	XMQIOHDR_UMR	1
		5	PROCESS	CHAR	8	XMQIOHDR_PROCESS	1
		6	EVENT	CHAR	8		
		7	DOC_FIELDS	INTEGER	4		
		8	REPORT	INTEGER	4		
		9	EXPIRY	INTEGER	4		
		10	FEEDBACK	INTEGER	4		
		11	ENCODING	INTEGER	4		
		12	CCSID	INTEGER	4		
		13	FORMAT	CHAR	8		
		14	PRIORITY	INTEGER	4		
		15	PERSISTENCE	INTEGER	4		
		16	MSGID	CHAR	24		
		17	CORRELID	CHAR	24		
		18	BACKOUT	INTEGER	4		
		19	USERID	CHAR	12		
		20	ACCOUNTING	CHAR	32		
		21	APPLIDENT	CHAR	32		
		22	PUTAPPLTYPE	INTEGER	4		
		23	PUTAPPLNAME	CHAR	28		
		24	PUTDATE	CHAR	8		
		25	PUTTIME	CHAR	8		
		26	APPLORIGIN	CHAR	4		
		27	OPTIONSID	CHAR	4		
		28	OPTIONS	INTEGER	4		
		29	WAITINTERVAL	INTEGER	4		
		30	REPLYTOQ	VARCHAR	48		
		31	REPLYTOQMGR	VARCHAR	48		
		32	RESOLVEDQ	VARCHAR	48		
		33	RESOLVEDQMGR	VARCHAR	48		

Table 3. DB2 Table Overview (continued)

Table	Recl	Seq	Column	Type	Length	Indexes	Seq
MQIOHDR_TEMP	512	1	PKEY	TIMESTMP	10		
		2	EKEY	TIMESTMP	10		
		3	MTYPE	CHAR	8		
		4	UMR	CHAR	28		
		5	PROCESS	CHAR	8	XMQIOHDRT_PROCESS	1
		6	EVENT	CHAR	8		
		7	DOC_FIELDS	INTEGER	4		
		8	REPORT	INTEGER	4		
		9	EXPIRY	INTEGER	4		
		10	FEEDBACK	INTEGER	4		
		11	ENCODING	INTEGER	4		
		12	CCSID	INTEGER	4		
		13	FORMAT	CHAR	8		
		14	PRIORITY	INTEGER	4		
		15	PERSISTENCE	INTEGER	4		
		16	MSGID	CHAR	24		
		17	CORRELID	CHAR	24		
		18	BACKOUT	INTEGER	4		
		19	USERID	CHAR	12		
		20	ACCOUNTING	CHAR	32		
		21	APPLIDENT	CHAR	32		
		22	PUTAPPLTYPE	INTEGER	4		
		23	PUTAPPLNAME	CHAR	28		
		24	PUTDATE	CHAR	8		
		25	PUTTIME	CHAR	8		
		26	APPLORIGIN	CHAR	4		
		27	OPTIONSID	CHAR	4		
		28	OPTIONS	INTEGER	4		
		29	WAITINTERVAL	INTEGER	4		
		30	REPLYTOQ	VARCHAR	48		
		31	REPLYTOQMGR	VARCHAR	48		
		32	RESOLVEDQ	VARCHAR	48		
		33	RESOLVEDQMGR	VARCHAR	48		
QUEDEL	68	1	PKEY	TIMESTMP	10	XQUEDEL	1
		2	EKEY	TIMESTMP	10	XQUEDEL_EKEY	1
		3	QUEUE	CHAR	8		
		4	QSN	INTEGER	4		
		5	UMR	CHAR	28		
QUEDFLD	294	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XQUEDFLD	1
		3	FIELD_ID	CHAR	8		
		4	FIELD_DATA	VARCHAR	256		
QUEPFLD	294	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XQUEPFLD	1
		3	FIELD_ID	CHAR	8		
		4	FIELD_DATA	VARCHAR	256		
QUEPUT	68	1	PKEY	TIMESTMP	10	XQUEPUT	1
		2	EKEY	TIMESTMP	10	XQUEPUT_EKEY	1
		3	QUEUE	CHAR	8		
		4	QSN	INTEGER	4		
		5	UMR	CHAR	28		
TLXIDOC	22020	1	PKEY	TIMESTMP	10	XTLXIDOC	1
		2	APDU	VARCHAR	22000		

Table 3. DB2 Table Overview (continued)

Table	Recl	Seq	Column	Type	Length	Indexes	Seq
TLXIFLD	289	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XTLXIFLD	1
		3	FIELD_ID	CHAR	3		
		4	FIELD_DATA	VARCHAR	256		
TLXIHDR	104	1	PKEY	TIMESTMP	10	XTLXIHDR	1
		2	EKEY	TIMESTMP	10		
		3	DKEY	TIMESTMP	10		
		4	UMR	CHAR	28		
		5	APDU_ID	CHAR	6		
		6	STATION	CHAR	8		
		7	SESSION	CHAR	4		
		8	SEQUENCE	CHAR	4		
		9	O_SESSION	CHAR	4		
		10	O_SEQUENCE	CHAR	4		
		11	ACK_ID	CHAR	3		
		12	ACK_REASON	CHAR	4		
		13	EXCEPTION	CHAR	1		
TLXODOC	22020	1	PKEY	TIMESTMP	10	XTLXODOC	1
		2	APDU	VARCHAR	22000		
TLXOFLD	289	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XTLXOFLD	1
		3	FIELD_ID	CHAR	3		
		4	FIELD_DATA	VARCHAR	256		
TLXOHDR	104	1	PKEY	TIMESTMP	10	XTLXOHDR	1
		2	EKEY	TIMESTMP	10		
		3	DKEY	TIMESTMP	10		
		4	UMR	CHAR	28		
		5	APDU_ID	CHAR	6		
		6	STATION	CHAR	8		
		7	SESSION	CHAR	4		
		8	SEQUENCE	CHAR	4		
		9	O_SESSION	CHAR	4		
		10	O_SEQUENCE	CHAR	4		
		11	ACK_ID	CHAR	3		
		12	ACK_REASON	CHAR	4		
		13	EXCEPTION	CHAR	1		
TLXOXIP	111	1	PKEY	TIMESTMP	10		
		2	AKEY	TIMESTMP	10	XTLXOXIP	1
		3	TXNUMR	CHAR	20		
		4	TXISEQ	CHAR	5		
		5	TXCOUNT	CHAR	5		
		6	TXATIME	CHAR	12		
		7	TXETIME	CHAR	12		
		8	TXSTATUS	CHAR	2		
		9	TXSLINE	CHAR	1		
		10	TXDURM	CHAR	5		
		11	TXDURS	CHAR	2		
		12	TRLINE	CHAR	2		
		13	TRTIME	CHAR	12		
		14	TXIP_STATUS	CHAR	5		

Appendix B. QMF Queries

All QMF sample queries are listed in this appendix. All arguments, except amounts, are strings and must therefore be supplied in quotes. Except for time and date arguments string arguments can specify SQL wildcard characters.

Table 4. QMF Queries

Query	Description	Arguments	Tables used	Results
IMRQ01A FIN Input cross reference	FIN input user-to-user messages showing their MIR, MOR and MUR. If the message does not contain a MUR, the TRN is shown instead. If a delivery notification was received for the message, the MOR is shown completed with the session and sequence number of the MT0.		FINIHDR FINIACK FINOHDR FINOMOR FINOMIR	Sending time ACK/NAK indicator Message type MIR w/o date MOR w/o date MUR or TRN
IMRQ01B FIN output cross reference	FIN output user-to-user messages showing their MIR, MOR and MUR.		FINOHDR FINOACK	Receive time ACK/NAK indicator Message type MIR w/o date MOR w/o date MUR or TRN
IMRQ02A Search FIN input messages	A single message or a range of messages sent. Correlated APDU's are listed additionally.	Correspondent MIR MUR Message type FromDate ToDate FromTime ToTime	FINIHDR FINIACK FINIDOC FINOHDR FINOMIR	Correspondent Session number ISN S.W.I.F.T. input date (MERVA) S.W.I.F.T. input time (MERVA) ACK/NAK date (MERVA) ACK/NAK time (MERVA) ACK/NAK indicator, ACK=0, NAK=1 Message type of correlated APDU Receive date of corr. APDU Receive time of corr. APDU Related MIR contained in corr. APDU
IMRQ02B Search FIN output messages	A single message or a range of messages received.	Correspondent Sender's MUR Message type FromDate ToDate FromTime ToTime	FINOHDR FINOACK FINODOC	Sender Sender's session number Sender's ISN Sender's MUR Message type SWIFT outdate SWIFT outtime SWIFT intime MERVA intime MERVA acktime SWIFT acktime ACK/NAK indicator, ACK=0 NAK=1

Table 4. QMF Queries (continued)

Query	Description	Arguments	Tables used	Results
IMRQ03A Amounts sent	All amounts sent to other correspondents. The amounts are grouped by currency and the totals are calculated per currency	Correspondent Message type Currency Low Amount High Amount FromDate ToDate FromTime ToTime	FINIHDR FINIF32	Destination (receiver) Sending date Sending time TRN Currency Amount Value date
IMRQ03B Amounts received	All amounts received from other correspondents. The amounts are grouped by currency and the totals are calculated per currency.	Correspondent Message type Currency Low Amount High Amount FromDate ToDate FromTime ToTime	FINOHDR FINOF32	Sender Receive date Receive time TRN Currency Amount Value date
IMRQ04A GPA APDU list	All S.W.I.F.T. GPA application protocol data units sent and received.		GPAIHDR GPAIACK GPAIDOC GPAOHDR GPAOACK GPAODOC	Send/receive date Send/receive time APDU
IMRQ04B FIN APDU LIST	All S.W.I.F.T. FIN application protocol data units sent and received.		FINIHDR FINIACK FINIDOC FINOHDR FINOACK FINODOC	Send/receive date Send/receive time APDU
IMRQ04C GPA APDU overview	All S.W.I.F.T. GPA messages sent and received with MIR and MOR.		GPAIHDR GPAIACK GPAOHDR GPAOACK	Send/receive date Send/receive time Application and APDU identifier Message type or ACK/NAK indicator, ACK=0 NAK=1 MIR without date MOR without date

Table 4. QMF Queries (continued)

Query	Description	Arguments	Tables used	Results
IMRQ04D FIN APDU overview	All S.W.I.F.T. FIN messages with MIR, MOR, and MUR. If a message contains no MUR the TRN is shown.		FINIHDR FINIACK FINOHDR FINOACK	Send/receive date Send/receive time Application and APDU identifier Message type or ACK/NAK indicator, ACK=0 NAK=1 MIR without date MOR without date MUR or TRN
IMRQ05A MERVA queue trace for UMR range	The routing of messages within MERVA. The messages will be displayed in chronological order, sorted by UMR sequence number.	LowUmrSeq HighUmrSeq	QUEPUT QUEDEL	PUT/DEL indicator PUT/DEL date PUT/DEL time UMR identifier UMR sequence number UMR date UMR time Queue name Queue sequence number
IMRQ05B MERVA queue trace for date range	The routing of messages within MERVA. The messages will be displayed in chronological order, sorted by UMR sequence number.	FromDate ToDate	QUEPUT QUEDEL	PUT/DEL indicator PUT/DEL date PUT/DEL time UMR identifier UMR sequence number UMR date UMR time Queue name Queue sequence number
IMRQ05C MERVA queue trace for time range	The routing of messages within MERVA during a specified time range of the current date. The messages will be displayed in chronological order, sorted by UMR sequence number.	FromTime ToTime	QUEPUT QUEDEL	PUT/DEL indicator PUT/DEL date PUT/DEL time UMR identifier UMR sequence number UMR date UMR time Queue name Queue sequence number

Table 4. QMF Queries (continued)

Query	Description	Arguments	Tables used	Results
IMRQ05D MERVA queue trace for date/time range	The routing of messages within MERVA during a specified time range of a specific date range (day). The messages will be displayed in chronological order, sorted by UMR sequence number.	FromDate ToDate FromTime ToTime	QUEPUT QUEDEL	PUT/DEL indicator PUT/DEL date PUT/DEL time UMR identifier UMR sequence number UMR date UMR time Queue name Queue sequence number
IMRQ05E MERVA queue trace for single ISN	The routing of a single message within MERVA.	ISN	QUEPUT QUEDEL FINIHDR	PUT/DEL indicator PUT/DEL date PUT/DEL time Session number ISN Queue name Queue sequence number
IMRQ05F MERVA queue trace for ISN range	The routing of messages within MERVA for a range of ISN's.	LowISN HighISN	QUEPUT QUEDEL FINIHDR	PUT/DEL indicator PUT/DEL date PUT/DEL time Session number ISN Queue name Queue sequence number
IMRQ05G MERVA queue trace for single MUR/TRN	The routing of a message within MERVA.	MUR/TRN	QUEPUT QUEDEL FINIHDR	PUT/DEL indicator PUT/DEL date PUT/DEL time Session number ISN Queue name Queue sequence number MUR (TRN, if MUR not available)

Table 4. QMF Queries (continued)

Query	Description	Arguments	Tables used	Results
IMRQ06A Undelivered message report (no MT011)	All messages requesting a delivery notification without receiving one. In addition all messages retrieved by newest reports MT066, MT082 and MT083 are listed if they have not received a delivery notification in the meantime. The messages will be displayed in chronological order.		FINIHDR FINIDOC FINOHDR FINOMIR	Date of input to S.W.I.F.T. Time of input to S.W.I.F.T. Message type Sender Receiver Session number ISN MUR Priority Obsolescence period
IMRQ06B Message w/o req. MT010/011, not NAK'ed	All messages that have not received a requested delivery notification or non-delivery warning. The messages will be displayed in chronological order.		FINIHDR FINIACK FINOHDR FINOMIR	Date of input to S.W.I.F.T. Time of input to S.W.I.F.T. Session number ISN Message type Receiver Delivery indicator Obsolescence period MUR ACK/NAK indicator, ACK=0 NAK=1
IMRQ07A OSN GAPs report	All OSN GAPs. **Note:** This query uses REXX procedures: • IMRX07AS • IMRX07AE • IMRX07AC • IMRX07AX		FINOHDR	Date of last message before gap Time of last message before gap Session no. of last message before gap Sequence no. of last message before gap Sequence no. of first message after gap Session no. of first message after gap Date of first message after gap Time of first message after gap
IMRQ07B ISN GAPs and duplicates report	All ISN GAPs. Duplicate messages have the same ISN, but a different session number. **Note:** This query uses REXX procedures: • IMRX07BS • IMRX07BE • IMRX07BC • IMRX07BX		FINIHDR	Date of last message before gap Time of last message before gap Session no. of last message before gap Sequence no. of last message before gap Sequence no. of first message after gap Session no. of first message after gap Date of first message after gap Time of first message after gap

Table 4. QMF Queries (continued)

Query	Description	Arguments	Tables used	Results
IMRQ07C Messages closing OSN GAPs	All messages possibly closing an OSN GAP. PDM message and retrieved message are listed.		FINOHDR FINOTRL	Receive date Receive time MOR Session number OSN Closing category (PDM or RTV)
IMRQ07D Messages closing ISN GAPs	All messages possibly closing an ISNn GAP. PDE messages are listed.		FINIHDR FINITRL	Sending date Sending time MIR Session number ISN Closing category (PDE)
IMRQ08A FIN retrieval APDU list MT020, MT021	All MT020 and MT021 with corresponding ACK/NAK info.		FINIHDR FINIDOC FINIACK FINOHDR FINODOC FINOACK	Time of message or ACK/NAK APDU of message or ACK/NAK
IMRQ08B FIN retrieval overview	All FIN MT020/MT021 and retrieved messages by MIR, MOR and MUR. If no MUR is available, the TRN is listed.		FINIHDR FINOHDR	Time of MT020/MT021 Message type MIR without date MOR with PUT date MUR or TRN of retrieved message
IMRQ08C Open single MOR retrievals	All FIN MT020 requesting single MOR's that were not yet received. (Synonym destinations are not supported.)		FINIHDR FINIACK FINIMOR FINOHDR	Time of MT020 Message type MIR of MT020 without date MOR of MT020 (destination) Requested MOR
IMRQ08D Open/incomplete retrieval requests	All FIN MT020 messages that have not yet received a requested report or received a multisection report with sections missing.		FINIHDR FINIDOC FINOHDR FINOTRL FINOSEC	Time of MT020 APDU of MT020
IMRQ09A FIN output PDE/PDM APDU list	All output messages with PDE/PDM trailer and the corresponding original message. They are stored with the referring PDE/PDM message.		FINOHDR FINODOC FINOTRL	Time of PDE/PDM or original message Category (p=PDE/PDM message, o=original message) APDU

Table 4. QMF Queries (continued)

Query	Description	Arguments	Tables used	Results
IMRQ09B FIN output PDE/PDM overview	All output messages with PDE/PDM trailer and the corresponding original message. They are stored with the referring PDE/PDM message.		FINOHDR FINOTRL	Time of PDE/PDM or original message Category (p=PDE/PDM message, o=original message) Message type MIR without date MOR without date MUR/TRN
IMRQ10A FIN input PDE APDU list	All input messages with PDE trailer and the corresponding original message. They are stored with the referring PDE message		FINIHDR FINIDOC FINITRL	Time of PDE or original message Category (p=PDE/PDM message, o=original message) APDU
IMRQ10B FIN input PDE overview	All input messages with PDE trailer and the corresponding original message by the MIR, MOR, and MUR or TRN. Original messages are stored with the referring PDE message.		FINIHDR FINITRL	Time of PDE or original message Category (p=PDE/PDM message, o=original message) Message type MIR without date MOR without date MUR/TRN
IMRQ11 FIN nomatch APDU list	FIN message without ACK/NAK. ACK's and NAK's without message MT010/011/015/019/039/059 without existing referred message		FINIHDR FINIACK FINIDOC FINIMIR FINOHDR FINOACK FINODOC FINOMIR	Time of message APDU
IMRQ12A FIN input can/rej NAK'ed APDU list	FIN input message that received a NAK, MT015, MT019, or MT059. The received message (NAK,..) is stored with the original message.		FINIHDR FINIACK FINIDOC FINOHDR FINODOC FINOMIR	Sending time APDU

Table 4. QMF Queries (continued)

Query	Description	Arguments	Tables used	Results
IMRQ12B FIN input canc./rej./NAK'ed overview	FIN input message that received a NAK, MT015, MT019, or MT059. The received message is stored with the original message. The messages are listed by their MIR, MOR, and MUR or/TRN.		FINIHDR FINIACK FINISER FINOHDR FINOSER FINOMIR	Time of negated message, MT015, MT019, or MT059 NAK error code for NAK'ed messages, reason for cancellation or rejection of MT015/019/059 Message type MIR without date MOR without date MUR/TRN
IMRQ13A NAK'ed GPA message APDU list	All NAK'ed GPA messages on input and output. The NAK is stored with the message.		GPAIHDR GPAIACK GPAIDOC GPAOHDR GPAOACK GPAODOC	Time of message or NAK APDU
IMRQ13B NAK'ed FIN message APDU list	All NAK'ed FIN messages on input and output. The NAK is stored with the message.		FINIHDR FINIACK FINIDOC FINOHDR FINOACK FINODOC	Time of message or NAK APDU
IMRQ13C NAK'ed GPA message overview	All NAK'ed GPA messages on input and output by MIR and MOR. The NAK is stored with the message.		GPAIHDR GPAIACK GPAISER GPAOHDR GPAOACK GPAOSER	Time of message or NAK NAK error code Message type MIR without date MOR without date
IMRQ13D NAK'ed FIN message overview	All NAK'ed FIN messages on input and output by MIR, MOR, MUR/TRN. The NAK is stored with the message.		FINIHDR FINIACK FINISER FINOHDR FINOACK FINOSER	Time of message or NAK NAK error code Message type MIR without date MOR without date MUR/TRN
IMRQ14A Delayed FIN output message APDU list	All received FIN messages with DLM trailer.		FINOHDR FINODOC FINOTRL	Receive time APDU

Table 4. QMF Queries (continued)

Query	Description	Arguments	Tables used	Results
IMRQ14B Delayed FIN output message overview	All received FIN messages with DLM trailer by MIR, MOR, MUR/TRN.		FINOHDR FINOTRL	Receive time Message type MIR without date MOR without date MUR/TRN
IMRQ15A Session control message APDU list	All messages controlling FIN and GPA session.		GPAIHDR FINIHDR GPAIACK FINIACK GPAIDOC FINIDOC GPAOHDR FINOHDR GPAOACK FINOACK GPAODOC FINODOC	Receive/send time APDU
IMRQ15B Session control message overview	All messages controlling FIN and GPA session by MIR and MOR.		GPAIHDR FINIHDR GPAIACK FINIACK GPAOHDR FINOHDR GPAOACK FINOACK	Receive/send time Application and APDU identifier Message type MIR without date MOR without date

Table 4. QMF Queries (continued)

Query	Description	Arguments	Tables used	Results
IMRQ16 S.W.I.F.T. message charge report	All messages charged by S.W.I.F.T. The output is grouped as in the S.W.I.F.T. bill. Extracts data from the following messages: • APDU 01 • ’05 QUIT’ • ’03 SELECT’ LTC application messages and service messages other than QUIT and SELECT are not taken into account by S.W.I.F.T. **Note:** This query uses REXX procedures: • IMRAPULL • IMRBRSUM • IMRMSGPR	FromDate ToDate	FINIHDR FINOHDR GPAIHDR GPAOHDR	Session number of message Sequence number of message Applid APDU id Message type APDU length Priority indicator Delivery indicator Comment about message unit and priority charge
IMRQ20A List of journal records	All journal records with journal ID X’02’, X’03’, or X’16’		JRNPUT	Date journal entry Time journal entry Journal id Journal data
IMRQ31 Outgoing telexes by document number	All outgoing telexes with the specified TXIP document number.	Document no.	TLXIDOC TLXIHDR TLXOHDR TLXOXIP	Timestamp of event Outgoing telex
IMRQ32 Outgoing telexes by telex number	All outgoing telexes with the specified telex number.	Telex number	TLXIDOC TLXIHDR TLXIFLD	Timestamp of event Outgoing telex
IMRQ33 Outgoing telexes by correspondent address	All outgoing telexes with the specified correspondent address.	Correspondent	TLXIDOC TLXIHDR TLXIFLD	Timestamp of event Outgoing telex
IMRQ34 Outgoing telexes by reference	All outgoing telexes with the specified reference code.	Reference code	TLXIDOC TLXIHDR TLXIFLD	Timestamp of event Outgoing telex
IMRQ35 Outgoing telexes by address and testkey	All outgoing telexes with the specified address and testkey.	Address Testkey	TLXIDOC TLXIHDR TLXIFLD	Timestamp of event Outgoing telex

Table 4. QMF Queries (continued)

Query	Description	Arguments	Tables used	Results
IMRQ36 Incoming telexes by document number	All incoming telexes with the specified TXIP document number.	Document no.	TLXODOC TLXOHDR TLXOXIP	Timestamp of event Incoming telex
IMRQ37 Incoming telexes by receiving department	All incoming telexes with the specified receiving department. It is assumed that the receiving department is contained in the first n lines of the telex. The value of n is specified in the telex scanner control table during customization: IMRTXS SECTION=(OUTOUT,TELEX) IMRTXS LIN=(1,N),FID='BEG'	Department	TLXODOC TLXOHDR TLXOFLD	Timestamp of event Incoming telex
IMRQ38 Incoming telexes by correspondent and testkey	All incoming telexes with the specified correspondent and testkey. It is assumed that the testkey and the correspondent name are contained in the first n lines of the telex. n is specified in the telex scanner control table during customization: IMRTXS SECTION=(OUTOUT,TELEX) IMRTXS LIN=(1,N),FID='BEG'	Correspondent Testkey	TLXODOC TLXOHDR TLXOFLD	Timestamp of event Incoming telex

Appendix C. Operator Messages

IMR000I Traffic Reconciliation not activated

Explanation: The Traffic Reconciliation program IMRICON was started, but Traffic Reconciliation was not activated in the MERVA ESA parameter module DSLPRM.

User Response: Code RECON=(YES,CONT) or RECON=(YES,STOP) in the MERVA ESA parameter module DSLPRM.

Module: IMRICON

IMR001E Module *mod* not found

Explanation: During startup of Traffic Reconciliation, the module *mod* was not found in the allocated libraries.

User Response: Check the concatenation of the allocated MERVA ESA and Traffic Reconciliation load libraries.

Module: IMRICON

IMR002E Function *func* was not found in MERVA ESA table *table*

Explanation: The function *func*, which is defined in the parameters module IMRPRM with parameter DUMQUE, is not defined in the active MERVA ESA function table *table*.

User Response: Check the function table *table* and the parameter module IMRPRM.

Module: IMRICON

IMR003E Transaction code is missing in function *func*

Explanation: The function *func*, which is specified by the DUMQUE parameter in the parameter module IMRPRM, does not specify a transaction to start the program IMRINSP.

User Response: Check the function table and the IMS/CICS definitions for the program IMRINSP.

Module: IMRICON

IMR004E *service request*, rc=*X'rc'*, rsn=*X'rsn'*

Explanation: A *request* call to a MERVA ESA service module *service* failed with return code *rc* and reason code *rsn*. The return and reason codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRICON

IMR005I IMRICON requests termination of MERVA ESA

Explanation: The Traffic Reconciliation nucleus program IMRICON detected errors during processing, and RECON=(YES,STOP) is specified in the MERVA ESA parameter module DSLPRM. IMRICON signals to the calling program DSLNUC to terminate processing of MERVA for ESA.

Module: IMRICON

IMR006I IMRICON is terminating

Explanation: The Traffic Reconciliation nucleus program IMRICON detected errors during processing, and RECON=(YES,CONT) is specified in the MERVA ESA parameter module DSLPRM. IMRICON terminates but MERVA ESA continues processing.

Module: IMRICON

IMR007I *n monitor* event file records processed

Explanation: *n* events generated by the Traffic Reconciliation monitor *monitor* have been inserted into the DB2 tables from the flip-flop data set.

The message is issued at IMRICON termination for any monitors defined in IMRPRM, and reflects the number of events inserted since IMRICON was last started.

System Action: None.

User Response: None. Information only.

Module: IMRICON

IMR010I IMREXTP requests termination of MERVA ESA

Explanation: The Traffic Reconciliation event extraction facility detected errors during processing, and RECON=(YES,STOP) is specified in the MERVA ESA parameter module DSLPRM. IMREXTP signals to the calling program DSLNUC to terminate processing of MERVA ESA.

Module: IMREXTP

IMR011I IMREXTP is terminating

Explanation: The Traffic Reconciliation event extraction program IMREXTP detected errors during processing, and RECON=(YES,CONT) is specified in the MERVA ESA parameter module DSLPRM. IMREXTP terminates but MERVA ESA continues processing.

User Response: To reactivate Traffic Reconciliation, terminate MERVA ESA, correct the error, and restart MERVA ESA.

Module: IMREXTP

IMR012E DSLSRVP *request***, module=***mod***, rc=***X′rc′***, rsn=***X′rsn′*

Explanation: A LOAD or RELEASE request for module *mod* failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREXTP

IMR013E *module* **not in DSLNTRT nucleus task server request table**

Explanation: The program *module* is a Traffic Reconciliation central service, but is not in the MERVA ESA nucleus task server request table.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREXTP

IMR015I Traffic Reconciliation Version 4.1.0 active

Explanation: Traffic Reconciliation is now active, and MERVA events will be captured.

User Response: None.

Module: IMREXTP

IMR016E Main storage could not be acquired

Explanation: Main storage available to the program was exhausted.

System Action: The program terminates.

User Response: Allocate more storage to the program.

Module: IMREXTP, IMRJRNX, IMRQUEX

IMR017E IMREXTP internal inconsistency

Explanation: The connection between the MERVA ESA nucleus and Traffic Reconciliation is inconsistent.

System Action: The program terminates.

User Response: Report the problem to IBM.

Module: IMREXTP

IMR020I Control and Event files initialized

Explanation: The system detected that the VSAM control data set was not initialized. Initialization is performed.

User Response: None.

Module: IMREXTF

IMR021I Event file *ddname* **selected**

Explanation: The system selected the file *ddname* for writing the events.

User Response: None.

Module: IMREXTF

IMR022I Event file swapping from *file1* **to** *file2*

Explanation: The system switched from *file1* to *file2* for writing the event records because *file1* was full.

User Response: None.

Module: IMREXTF

IMR023I Event files are full

Explanation: The flip-flop writer IMREXTF tried to switch to the other event file but it was not yet completely processed. Depending on the value of the RECON parameter in DSLPRM, either MERVA ESA processing continues (if RECON=(YES,CONT)) or terminates (if RECON=(YES,STOP)).

User Response: If MERVA ESA terminates (RECON=(YES,STOP)), start the event insertion program IMRINSP. After completion, restart MERVA ESA.

If RECON=(YES,CONT) was specified, MERVA ESA continues processing but reconciliation events are lost. To reactivate the logging of events, terminate MERVA ESA and proceed as described in the previous paragraph.

Module: IMREXTF

IMR024I Event file stats: put=*num1***, wait=***num2*

Explanation: This message is issued during Traffic Reconciliation termination by the flip-flop writer program IMREXTF. During event extraction, *num1* put requests occurred. The program had to wait for completion *num2* times (VSAM CHECK).

If the proportion of waits is low, the effect of Traffic Reconciliation VSAM I/O overhead on the MERVA ESA nucleus is low.

User Response: None.

Module: IMREXTF

IMR025E ENQ Error, rsn=*X′rsn′*

Explanation: The system issued an ENQ request that failed with reason code *rsn*. The flip-flop CTL file cannot be updated.

The event insertion program may not be able to insert the last events until reconciliation is restarted and the CTL file successfully enqueued and updated.

User Response: See the description of the MVS ENQ macro in *OS/390 MVS Assembler Services Reference* for an explanation.

Module: IMREXTF

IMR026E VSAM *request*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: The system issued a SHOWCB or MODCB request that failed with return code *rc* and reason code *rsn*. The return and reason codes are given in hexadecimal.

User Response: See *DFSMS/MVS Macro Instructions for Data Sets* for an explanation.

Module: IMREXTF

IMR027E VSAM *request*, **DDNAME=***name*, **rc=***rc*

Explanation: The system issued an OPEN or CLOSE request for the VSAM data set with ddname *name*, which failed with return code *rc*. The return code is given in hexadecimal.

User Response: See *DFSMS/MVS Macro Instructions for Data Sets* for an explanation.

Module: IMREXTF

IMR028E VSAM *request*, **DDNAME=***name*, **fdbk=***fdbk*

Explanation: The system issued a processing request *request* that failed with feedback code *fdbk*. The feedback code is given in hexadecimal.

User Response: See *DFSMS/MVS Macro Instructions for Data Sets* for an explanation.

Module: IMREXTF

IMR030E DSLSRVP *request*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREXTF

IMR031W Event file threshold reached: *load* **full**

Explanation: The percentage *load* of both VSAM flip-flop data sets taken together contains events for DB2 insertion. This value exceeds the threshold value specified by the FFTHRESH parameter in IMRPRM.

User Response: Start the insertion transaction to process these events.

Module: IMREXTF

IMR032I Event file space reused, now *load* **full**

Explanation: Due to swapping of the VSAM flip-flop, the load was reduced to the percentage *load* that is below the threshold value specified in IMRPRM.

User Response: None.

Module: IMREXTF

IMR050E DSLSRVP *request*, **module=***mod*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A LOAD or RELEASE request for module *mod* failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRINSP

IMR051E DSLSRVP *request*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRINSP

IMR054E ASMTDLI request=*req*, **status=***st*

Explanation: An IMS request *req* failed with status code *st*.

User Response: See *IMS/ESA V5 Application Programming: Transaction Manager* for an explanation.

Module: IMRINSP

IMR055E SQL table=*table*, **request=***req*, **SQLCODE=***rc*

Explanation: An SQL statement *req* against the table *table* failed with SQLCODE *rc*. The insertion transaction or program terminates.

User Response: See *DB2 for OS/390 Messages and Codes* for an explanation.

Module: IMRINSP

IMR056W MERVA ESA must not be active (DSLNIC RC=*rc*)

Explanation: The batch version of the event insertion program (IMRINSP) was started, but MERVA ESA is active. To avoid conflicts between the online and batch versions of IMRINSP, the batch version will only run when MERVA ESA is inactive. The DSLNIC ALLOC return code is returned and will normally be 0.

User Response: To insert flip-flop events into the DB2 tables while MERVA ESA is active initiate the insertion transaction. You can normally do this by starting the insertion transaction controller (IMRICON) from the MERVA CMD panel with the command `S IMRICON`.

Module: IMRINSP

IMR058I Control database IMRCTL is empty

Explanation: During startup the program detected an empty IMRCTL table. Therefore, restarting can only be based on values in the flip-flop control file, which has the ddname IMRCTL. If the system terminated abnormally and the control database was deleted and re-created, events up to the value of the COMMIT parameter of the parameter module IMRPRM can be processed again, and therefore can be duplicated in the DB2 tables.

User Response: None.

Module: IMRINSP

IMR059E EXEC CICS *request*, EIBRCODE=*rc*

Explanation: An EXEC CICS request *request* failed with return code *rc*.

request can have the following values and meanings:

EXTR	EXTRACT EXIT PROGRAM
FREE	FREE
INQE	INQUIRE EXIT PROGRAM
INQR	INQUIRE SYSTEM RELEASE
RCVE	RECEIVE
RTRV	RETRIEVE

User Response: See *CICS/ESA V4R1 Application Programming Reference* for an explanation.

Module: IMRINSP

IMR060E DB2 connection unavailable

Explanation: The check for the availability of DB2 failed.

User Response: Ensure that the CICS attachment facility was started.

Module: IMRINSP

IMR061I *n monitor* events processed by IMRI

Explanation: *n* events generated by Traffic Reconciliation monitor *monitor* have been inserted into the DB2 tables from the flip-flop data set.

The message is issued when the insertion program (IMRINSP) terminates for any monitors defined in IMRPRM, and reports the number of events inserted since IMRINSP was last started.

When running as a transaction, IMRINSP issues this message only if IMRICON is inactive. Otherwise IMRICON maintains these counts and issues a similar message (IMR007I) when it terminates.

System Action: None.

User Response: None. Information only.

Module: IMRINSP

IMR062E Program PARM invalid

Explanation: A value in the JCL EXEC PARM parameter is invalid.

System Action: The program terminates.

User Response: Correct the parameter.

Module: IMRINSP

IMR070E VSAM *request*, DDNAME=*name*, rc=*X'rc'*

Explanation: The system issued an OPEN or CLOSE request for the VSAM data set with ddname *name*, which failed with return code *rc*. The return code is given in hexadecimal.

User Response: See *DFSMS/MVS Macro Instructions for Data Sets* for an explanation.

Module: IMRINSF

IMR071E VSAM *request*, DDNAME=*name*, fdbk=*X'fdbk'*

Explanation: The system issued a processing request *request* that failed with feedback code *fdbk*. The feedback code is given in hexadecimal.

User Response: See *DFSMS/MVS Macro Instructions for Data Sets* for an explanation.

Module: IMRINSF

IMR072E ENQ Error, rs(X) = *rsn*

Explanation: The program IMREXTF issued an ENQ request that failed with reason code *rs*. The reason code is given in hexadecimal.

User Response: See *OS/390 MVS Assembler Services Reference* for an explanation.

Module: IMRINSF

IMR073E DSLSRVP *request*, **rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRINSF

IMR074E Event file control data is inconsistent

Explanation: Checking the VSAM control data set, which has the ddname IMRCTL, failed.

User Response: Check the VSAM cluster allocated to ddname IMRCTL.

If the control data set was destroyed, do the following:

1. Delete and redefine the VSAM clusters allocated to IMRCTL, IMRDSA, and IMRDSB. Events may be lost.
2. Delete the row in the control table IMRCTL corresponding to the MERVA ESA system using these files.
3. Start MERVA ESA.

Module: IMRINSF

IMR075E VSAM *request*, **rc=***rc*, **rsn=***rsn*

Explanation: The system issued a SHOWCB or MODCB request that failed with return code *rc* and reason code *rsn*. The return and reason codes are given in hexadecimal.

User Response: See *DFSMS/MVS Macro Instructions for Data Sets* for an explanation.

Module: IMRINSF

IMR076I Event file control data set IMRCTL is empty

Explanation: During startup of the insertion program, the system detected an empty control file allocated to ddname IMRCTL. Insertion in the databases was not started.

User Response: Start MERVA ESA with Traffic Reconciliation activated. Traffic Reconciliation will initialize the VSAM clusters.

Module: IMRINSF

IMR077I Insertion starts from *file* **after RBA=***rba*

Explanation: The insertion transaction or program inserting events into the databases starts processing events in the VSAM data set allocated to ddname *file* at the first record after the relative byte address *rba*.

This message is also issued when processing switches from one flip-flop file to the other.

User Response: None.

Module: IMRINSF

IMR100I Queue monitor started

Explanation: The queue monitor extraction program is now active.

User Response: None.

Module: IMRQUEX

IMR101I Queue monitor terminated

Explanation: The extraction facility of the queue monitor terminated.

User Response: None.

Module: IMRQUEX

IMR102E DSLSRVP *request*, **rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRQUEX

IMR103E DSLMMFS *request*, **rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: A *request* call to the serving module DSLMMFS failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRQUEX

IMR150E SQL table=*table*, **request=***req*, **SQLCODE=***rc*

Explanation: An SQL statement *req* against the table *table* failed with SQLCODE *rc*.

User Response: See *DB2 for OS/390 Messages and Codes* for an explanation.

Module: IMRQUEI

IMR153E DSLSRVP *request*, **rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRQUEI

IMR200I SWIFT monitor started

Explanation: The event extraction facility of the S.W.I.F.T. monitor is active.

User Response: None.

Module: IMRSWFX

IMR201I SWIFT monitor terminated

Explanation: The extraction facility of the S.W.I.F.T. monitor terminated.

User Response: None.

Module: IMRSWFX

IMR202E DSLSRVP *request*, **rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSWFX

IMR203E DSLTOFSV *request*, **field=***fld*, **rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: A *request* call to the serving module DSLTOFSV for the field *fld* failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSWFX

IMR204E DSLMMFS *request*, **rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: A *request* call to the serving module DSLMMFS failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSWFX

IMR205E DSLSRVP *request*, **rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSWFX

IMR220E DSLSRVP error *rc* **on request** *req*

Explanation: A *req* call to the MERVA ESA services program DSLSRVP failed with return code *rc*.

System Action: The program terminates.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation of the code.

Module: IMRSWFF

IMR221E DSLSRVP error *rc* **while loading module** *mod*

Explanation: Loading of module *mod* failed.

System Action: The program terminates.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation of the code.

In CICS, the module must have been declared to CICS.

Module: IMRSWFF

IMR222E DSLTOFSV error *rc* **on request** *req*

Explanation: A call to the message field access service failed.

System Action: The program terminates.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation of the code.

The message causing the problem is the first in the queue assigned to transaction IMRSWFF. The easiest way to inspect it is to run the REXX "tofscan" sample program. See DSLBA14R in the *MERVA for ESA V4 Application Programming Interface Guide*.

Module: IMRSWFF

IMR223E DSLMMFS error *rc* **on request** *req* **for medium** *medium*

Explanation: A call to the message formatting service failed.

System Action: The program terminates.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation of the code.

Inspect the message causing the problem. This is the first message in the queue to which IMRSWFF is assigned.

Module: IMRSWFF

IMR224E DSLNICT error *rc* **on request** *req*

Explanation: A call to the intertask communication service failed.

System Action: The program terminates.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation of the return code.

Module: IMRSWFF

IMR225E DSLQMGT *request*, **rc** *rc*, **queue** *queue*

Explanation: A DSLQMGT *request* call on function *queue* failed with return code *rc*.

System Action: The program terminates.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation of the return code.

Module: IMRSWFF

IMR226E *status* **error on IMS DC call** *call*

Explanation: The IMS DC call *call* failed with status code *status*.

System Action: The program terminates.

User Response: Refer to *IMS/ESA V5 Application Programming: Transaction Manager* for the meaning of the status code.

Module: IMRSWFF

IMR227E *rc* **error on** *service* **macro**

Explanation: A call of the MVS service *service* failed.

System Action: The program terminates.

User Response: Refer to *OS/390 MVS Assembler Services Reference* for an explanation of the error.

Module: IMRSWFF

IMR228E IMRSWFF requires SWIFT monitor specification

Explanation: The Financial Message Capture transaction (IMRF, program IMRSWFF) was initiated, but the S.W.I.F.T. monitor was not specified in the Traffic Reconciliation parameters module (IMRPRM).

System Action: The transaction terminates.

User Response: Correct your IMRPRM.

Module: IMRSWFF

IMR250E SQL table=*table*, **request=***req*, **SQLCODE=***rc*

Explanation: An SQL statement *req* against the table *table* failed with SQLCODE *rc*.

User Response: See *DB2 for OS/390 Messages and Codes* for an explanation.

Module: IMRSWFD

IMR253E DSLSRVP *request*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation of the code.

Module: IMRSWFD

IMR300I Journal monitor started

Explanation: The event extraction facility of the journal monitor is active.

User Response: None.

Module: IMRJRNX

IMR301I Journal monitor terminated

Explanation: The event extraction facility of the journal monitor terminated.

User Response: None.

Module: IMRJRNX

IMR302E DSLSRVP *request*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRJRNX

IMR304E System macro=*name*, **rc=***X′rc′*

Explanation: A call of the MVS system service *name* failed with return code *rc*.

User Response: Refer to *OS/390 MVS Assembler Services Reference* for an explanation of the return code.

Module: IMRJRNX

IMR310E DSLSRVP *request* **error, rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A call to the MERVA service program failed.

System Action: The program terminates.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRRDU, IMRRDUF, IMRRDUI

IMR311E VSAM *request* **error, rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: The VSAM request *request* failed.

System Action: The program terminates.

User Response: Refer to *DFSMS/MVS Macro Instructions for Data Sets* for an explanation of the codes.

Module: IMRRDUF

IMR312E Module could not be loaded: *mod*

Explanation: The module *mod* is required by the delete utility, but is not in load module library concatenation.

System Action: The program terminates.

User Response: Correct the job's JCL. See "Chapter 9. Deleting Traffic Reconciliation Data" on page 89 for an example.

Module: IMRRDU

IMR313E Program PARM missing

Explanation: A **PARM=** statement must be provided on the EXEC statement.

System Action: The program terminates.

User Response: Refer to "Chapter 9. Deleting Traffic Reconciliation Data" on page 89 for a description of the parameters you can specify.

Module: IMRRDU

IMR314E Program PARM *parm* **invalid**

Explanation: The parameter *parm* in the **PARM=** string of the EXEC statement has an invalid format, is not numeric, or not recognized.

System Action: The program terminates.

User Response: Refer to "Chapter 9. Deleting Traffic Reconciliation Data" on page 89 for a description of the parameters you can specify.

Module: IMRRDU

IMR315E Program PARM DATE= or DAYS= required

Explanation: A delete date is required by the delete utility. Either **DATE=** or **DAYS=** must be provided in the EXEC statement **PARM=** parameter.

System Action: The program terminates.

User Response: Refer to "Chapter 9. Deleting Traffic Reconciliation Data" on page 89 for a description of the **DATE=** and **DAYS=** parameters.

Module: IMRRDU

IMR316E Program PARM monitor name required

Explanation: A Traffic Reconciliation monitor name is required by the delete utility, but was not found in the job's **PARM=** parameter.

System Action: The program terminates.

User Response: Refer to "Chapter 9. Deleting Traffic Reconciliation Data" on page 89 for a description of the **PARM=** parameters.

Module: IMRRDU

IMR317E Program PARM more than 1 monitor specified

Explanation: In any one execution of the delete utility, only one Traffic Reconciliation monitor name can be specified.

System Action: The program terminates.

User Response: Correct the **PARM=** parameter. Refer to "Chapter 9. Deleting Traffic Reconciliation Data" on page 89 for details on running the delete utility.

Module: IMRRDU

IMR320E SQL error *rc*, **program** *prog*, **line** *line*

Explanation: An SQL statement failed with SQLCODE *rc*. The statement is in module *prog*, at line number *line*. Normally, additional information will be issued in the form of DB2 messages.

System Action: The program terminates.

User Response: Refer to *DB2 for OS/390 Messages and Codes* for an explanation of the SQL code and the follow-up DB2 messages.

Module: IMRRDUI, IMRRDUJ, IMRRDUM, INRRDUQ, IMRRDUS, IMRRDUT

IMR321E Table *table*, **column** *col* = *value* **missing**

Explanation: An expected row is missing from the table *table*. Column *col* with value *value* is the key with which the missing row was fetched. The database record is in an inconsistent state.

System Action: The program terminates.

User Response: Determine why the row is missing. Column *col* is probably a key column from an HDR table row. You will need to delete the database record by hand before the delete utility can be continued.

Module: IMRRDUS, IMRRDUT

IMR350E SQL table=*table*, request=*req*, SQLCODE=*rc*

Explanation: An SQL statement *req* against the table *table* failed with SQLCODE *rc*.

User Response: See *DB2 for OS/390 Messages and Codes* for an explanation.

Module: IMRJRNI

IMR353E DSLSRVP *request*, rc=*X'rc'*, rsn=*X'rsn'*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRJRNI

IMR400I Telex monitor started

Explanation: The telex monitor was started, and telex events will be logged in the telex monitor DB2 tables.

User Response: None.

Module: IMRTLXX

IMR401I Telex monitor terminated

Explanation: The telex monitor terminated. Logging of telex events stops if the MERVA for ESA nucleus also stops. Otherwise, the logging of events in DB2 tables can continue, because it is done by the independent transactions ENLR and ENLS.

If the logging in the DB2 tables should be stopped, the sending and receiving transactions ENLS and ENLR of MERVA ESA must be stopped.

User Response: None.

Module: IMRTLXX

IMR403E DSLSRVP *request*, rc=*X'rc'*, rsn=*X'rsn'*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRTLXX

IMR410E DSLSRVP *request*, module=*mod*, rc=*X'rc'*, rsn=*X'rsn'*

Explanation: A LOAD or RELEASE request for module *mod* failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRTLXP

IMR411E DSLSRVP *request*, rc=*X'rc'*, rsn=*X'rsn'*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRTLXP

IMR412E DSLMMFS *request*, rc=*X'rc'*, rsn=*X'rsn'*

Explanation: A *request* call to the serving module DSLMMFS failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRTLXP

IMR413E DSLMMFS request=*cmd*, rc=*X'rc'*, rsn=*X'rsn'*, MCB=*mcb*, Id=*id*

Explanation: A MERVA ESA MFS error occurred while trying to extract a message from the message TOF form using format *id* of MCB *mcb*.

System Action: The program terminates.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation of the DSLMMFS codes.

Module: IMRTLXP

IMR414E IMREXTF VSAM writer, rc=*X'rc'*

Explanation: An event record could not be written to the flip-flop data sets IMRDSA, IMRDSB, or IMRCTL.

System Action: The Telex event extraction program terminates.

User Response: An additional message should have been issued by IMREXTF. Correct the VSAM problem.

Module: IMRTLXP

IMR415E DSLNICT request=*service*, rc=*X'rc'*

Explanation: The MERVA ESA central service call to program *service* failed with return code *rc*.

System Action: The program terminates.

User Response: Check that program *service* is correctly defined to MERVA as a central service (table DSLNTRT), and that the program can be loaded by MERVA.

See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRTLXP

IMR450E DSLSRVP *request*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRTLXI

IMR453E SQL table=*table*, **request=***req*, **SQLCODE=***rc*

Explanation: An SQL statement *req* against the table *table* failed with SQLCODE *rc*.

User Response: See *DB2 for OS/390 Messages and Codes* for an explanation.

Module: IMRTLXI

IMR500I MLINK monitor for ASP=*asp* **started**

Explanation: A Traffic Reconciliation MLINK monitor was activated for the ASP called *asp*.

User Response: None.

Module: IMRMLKX

IMR501I MLINK monitors terminated

Explanation: The Traffic Reconciliation MLINK monitors were terminated during shutdown of MERVA for ESA.

User Response: None.

Module: IMRMLKX

IMR503E DSLSRVP *request*, **mod=***mod*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: The request *request* to the MERVA ESA service program (DSLSRVP) failed.

System Action: The program terminates.

User Response: If the request is **LOAD**, check that the load module *mod* can be loaded by MERVA.

Module: IMRMLKX

IMR504I MQI monitor for PROCESS=*process* **started**

Explanation: A Traffic Reconciliation MQI monitor started for the MERVA-MQI attachment process *process*.

System Action: The program terminates.

User Response: None. Information only.

Module: IMRMQIX

IMR505I MQI monitors terminated

Explanation: All MQI process monitors have been normally terminated.

System Action: The program terminates.

User Response: None. Information only.

Module: IMRMQIX

IMR506E DSLSRVP *request*, **mod=***mod*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: The request *request* to the MERVA ESA service program (DSLSRVP) failed.

System Action: The program terminates.

User Response: If the request is **LOAD**, check that the load module *mod* can be loaded by MERVA.

Module: IMRMQIX

IMR510E SQL table=*table*, **request=***req*, **SQLCODE=***rc*

Explanation: An SQL statement *req* against the table *table* failed with SQLCODE *rc*.

User Response: See *DB2 for OS/390 Messages and Codes* for an explanation.

Module: IMRMLKI

IMR511E DSLSRVP *request*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRMLKI

IMR520E DSLSRVP *request*, **module=***mod*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A LOAD or RELEASE request for module *mod* failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRMLKP

IMR521E DSLSRVP *request*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRMLKP

IMR522E DSLMMFS *request*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call to the serving module DSLMMFS failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRMLKP

IMR527E DSLTOFSV *request*, **field=***fld*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call to the serving module DSLTOFSV failed with return code *rc* and reason code *rsn* for field *fld*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRMLKP

IMR531E *string* **mapping: rc=***X′rc′*, **rsn=***X′rsn′*, **MCB=***mtype*, **Id=***id*

Explanation: String mapping for a MERVA Link event failed. *rc* and *rsn* are the DSLMMFS return and reason codes. *mtype* and *id* are the message and format identifiers derived from a *string* parameter in a MERVA Link monitor **EVENT** specification.

If *mtype* is blank, your **MTYPE** or **NET** specification did not yield a value.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRMLKP

IMR532E DSLNICT request=DSLQMGT, rc=*X′rc′*

Explanation: A DSLNICT request for DSLQMGT failed with return code *rc*. The code is given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRMLKP

IMR533E DSLQMGT UMR, rc=*X′rc′*, **rsn=***X′rsn′*

Explanation: A DSLQMGT TYPE=UMR request failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Traffic Reconciliation requires that **UMR=YES** is specified in your DSLPRM.

Module: IMRMLKP

IMR534E IMREXTF VSAM Writer, rc=*X′rc′*

Explanation: An event record could not be written to the flip-flop data sets.

System Action: The program terminates. The MERVA Link ASP being monitored is stopped if **RECON=(YES,STOP)** is specified in DSLPRM.

User Response: An additional message may have been issued by IMREXTF. Correct the problem with the VSAM data sets IMRDSA, IMRDSB, or IMRCTL.

Module: IMRMLKP

IMR570E DSLSRVP *request*, **mod=***mod*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A LOAD or RELEASE request for module *mod* failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

System Action: The program terminates.

User Response: If the request is **LOAD**, check that the load module *mod* can be loaded by MERVA.

Module: IMRMQIP

IMR571E DSLSRVP *request*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

System Action: The program terminates.

User Response: Refer to *MERVA for ESA V4 Messages and Codes* for an explanation of the codes.

Module: IMRMQIP

IMR572E DSLTOFSV request=*cmd*, **fld=***fld*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *req* request to the service module DSLTOFSV for field *fld* failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

System Action: The program terminates.

User Response: Refer to *MERVA for ESA V4 Messages and Codes* for an explanation of the codes.

Module: IMRMQIP

IMR573E **DSLMMFS NPUT, rc=***X'rc'***, rsn=***X'rsn'***, MCB=***mcb***, Id=***id*

Explanation: A TOF to NET mapping request using the MCB *mcb* and format identification *id* to the service module DSLMMFS failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

System Action: The program terminates.

User Response: Refer to *MERVA for ESA V4 Messages and Codes* for an explanation of the codes.

Module: IMRMQIP

IMR574E **DSLNICT** *request***, rc=***X'rc'*

Explanation: A DSLNICT request for DSLQMGT failed with return code *rc*. The code is given in hexadecimal.

System Action: The program terminates.

User Response: Refer to *MERVA for ESA V4 Messages and Codes* for an explanation of the code.

Module: IMRMQIP

IMR575E **DSLQMGT UMR, rc=***X'rc'***, rsn=***X'rsn'*

Explanation: A DSLQMGT TYPE=UMR request failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

System Action: The program terminates.

User Response: Refer to *MERVA for ESA V4 Messages and Codes* for an explanation of the codes.

Traffic Reconciliation requires that **UMR=YES** is specified in your DSLPRM.

Module: IMRMQIP

IMR576E **IMREXTF VSAM Writer, rc=***X'rc'*

Explanation: An event record could not be written to the flip-flop data sets.

System Action: The program terminates. The MERVA-MQI attachment process being monitored is stopped if **RECON=(YES,STOP)** is specified in your DSLPRM.

User Response: An additional message may have been issued by the IMREXTF nucleus central service. Correct the problem with the VSAM data set IMRDSA, IMRDSB, or IMRCTL.

Module: IMRMQIP

IMR580E **SQL table=***table***, request=***req***, SQLCODE=***rc*

Explanation: An SQL statement *req* against the table *table* failed with SQLCODE *rc*. Normally, additional information will be issued in the form of DB2 messages.

System Action: The program terminates.

User Response: Refer to *DB2 for OS/390 Messages and Codes* for an explanation of the SQL code and the follow-up DB2 messages.

Module: IMRMQII

IMR581E **DSLSRVP** *request mod***, rc=***X'rc'***, rsn=***X'rsn'*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

System Action: The program terminates.

User Response: If the request is **LOAD**, check that the load module *mod* can be loaded by MERVA.

Module: IMRMQII

IMR601E **DSLSRVP** *request***, module=***mod***, rc=***X'rc'***, rsn=***X'rsn'*

Explanation: A LOAD or RELEASE request for module *mod* failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRMD001

IMR602E **DSLTOFSV** *request***, field=***fld***, rc=***X'rc'***, rsn=***X'rsn'*

Explanation: A *request* call to the serving module DSLTOFSV failed with return code *rc* and reason code *rsn* for field *fld*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRMD001

IMR603W **No reconciliation class is assigned to you**

Explanation: Your MERVA user profile does not define a Traffic Reconciliation user class.

User Response: See your user profile administrator for assigning a user class to you.

Module: IMRMD001

IMR605W You are not allowed to access any query

Explanation: Traffic Reconciliation queries are defined for the system, but none of them serves the Traffic Reconciliation security class you are in.

User Response: None.

Module: IMRMD001

IMR611E DSLSRVP *request***, module=***mod***, rc=***X′rc′***, rsn=***X′rsn′*

Explanation: A LOAD or RELEASE request for module *mod* failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRMC011

IMR612E Target function not defined in function table

Explanation: The name of a MERVA function is invalid.

User Response: Enter the correct name of a MERVA function.

Module: IMRMC011

IMR613E Invalid maximum number of messages

Explanation: The field requires a number as the maximum number of messages to be processed.

User Response: Enter a correct number.

Module: IMRMC011

IMR614E Request type must be R(etrieval) or L(ist).

Explanation: Only the values R (for ″retrieval″) or L (for ″list″) are allowed.

User Response: Enter R (for ″retrieval″) or L (for ″list″).

Module: IMRMC011

IMR615E DSLTOFSV *request***, rc=***X′rc′***, rsn=***X′rsn′*

Explanation: A *request* call to the serving module DSLTOFSV failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRMC011

IMR616E Indicator must be R(oute) or Q(ueue)

Explanation: Only the values R (for ″route″) or Q (for ″queue″) are allowed.

User Response: Enter R (for ″route″) or Q (for ″queue″).

Module: IMRMC011

IMR617E Invalid number of messages to skip

Explanation: The field requires a number to specify the number of messages to be skipped.

User Response: Enter the correct number.

Module: IMRMC011

IMR631E DSLTOFSV *request***, rc=***X′rc′***, rsn=***X′rsn′*

Explanation: A *request* call to the serving module DSLTOFSV failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRMS031

IMR641E DSLTOFSV *request***, rc=***X′rc′***, rsn=***X′rsn′*

Explanation: A *request* call to the serving module DSLTOFSV failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREQRY

IMR642E DSLMMFS *request***, rc=***X′rc′***, rsn=***X′rsn′***, mcb=***mcb*

Explanation: A *request* call using the MCB *mcb* to the serving module DSLMMFS failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREQRY

IMR643E Command not known

Explanation: The command you issued is not known to the system.

User Response: Enter a valid MERVA command.

Module: IMREQRY

IMR644W You are not allowed to access query=*query*

Explanation: You tried to process the query *query* for which you lack authorization.

User Response: See your system administrator to get access to the query.

Module: IMREQRY

IMR645E DSLMMFS *request*, rc=*X'rc'*, rsn=*X'rsn'*

Explanation: A *request* call to the serving module DSLMMFS failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREQRY

IMR646E Command not allowed in this state

Explanation: The command issued is not allowed in this state of the system.

User Response: Enter a valid MERVA command.

Module: IMREQRY

IMR647E DSLNICT *request*, rc=*X'rc'*

Explanation: A *request* call for DSLNICT failed with return code *rc*. The code is given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREQRY

IMR648E DSLQMGT *request*, rc=*X'rc'*

Explanation: A *request* call for DSLQMGT failed with return code *rc*. The code is given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREQRY

IMR649E DSLSRVP *request*, module=*mod*, rc=*X'rc'*, rsn=*X'rsn'*

Explanation: A *request* call for module *mod* failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREQRY

IMR650E DSLSRVP *request*, rc=*X'rc'*, rsn=*X'rsn'*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREQRY

IMR651E DSLTOFSV *request*, rc=*X'rc'*, rsn=*X'rsn'*, field=*fld*

Explanation: A *request* call to the serving module DSLTOFSV failed with return code *rc* and reason code *rsn* for field *fld*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREQRY

IMR652E Query *query* not defined

Explanation: You tried to run a query that is not defined in the system.

User Response: Run one of the defined queries or contact your system administrator to add your query.

Module: IMREQRY

IMR653W No IMR user class is assigned to you

Explanation: For you to be able to run queries, your MERVA profile must specify a user class in the form IMR(nnn).

User Response: Define a user class in your MERVA profile using the USR functions.

Module: IMREQRY

IMR654E Command requires a valid query type

Explanation: You issued a command without a mandatory query type as parameter.

User Response: Reenter the command together with a query type.

Module: IMREQRY

IMR655I Query *query* queued and released for processing

Explanation: You finished entering a query *query*. The system accepted the query, queued it, and released it for processing.

User Response: Wait for the response that arrives after processing in the target queue.

Module: IMREQRY

IMR656W You are not allowed to access any query

Explanation: You returned to the MERVA function for running queries, but there are no queries that you are allowed to process.

User Response: Check your IMR user class.

Module: IMREQRY

IMR661E DSLTOFSV *request*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call to the serving module DSLTOFSV failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRECMD

IMR662E DSLMMFS *request*, **rc=***X′rc′*, **rsn=***X′rsn′*, **mcb=***mcb*

Explanation: A *request* call using the MCB *mcb* to the serving module DSLMMFS failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRECMD

IMR663W Command not known

Explanation: The command you issued is not known to the system.

User Response: Enter a valid MERVA command.

Module: IMRECMD

IMR664W Command only allowed in message selection state

Explanation: You tried to issue a command that is not valid in this state.

User Response: Change the state of the system and reissue the command.

Module: IMRECMD

IMR665E DSLMMFS *request*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call to the serving module DSLMMFS failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRECMD

IMR666W Command DELALL not allowed for this function

Explanation: The command you issued is not allowed in this function.

User Response: Enter a valid MERVA command.

Module: IMRECMD

IMR667E DSLNICT *request*, **rc=***X′rc′*

Explanation: A *request* call for DSLNICT failed with return code *rc*. The code is given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRECMD

IMR668E DSLQMGT *request*, **rc=***X′rc′*

Explanation: A *request* call for DSLQMGT failed with return code *rc*. The code is given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRECMD

IMR669E DSLSRVP *request*, **module=***mod*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call for module *mod* failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRECMD

IMR670E DSLSRVP *request*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRECMD

IMR671E DSLTOFSV *request*, **rc=***X′rc′*, **rsn=***X′rsn′*, **field=***fld*

Explanation: A *request* call to the serving module DSLTOFSV failed with return code *rc* and reason code *rsn* for field *fld*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRECMD

IMR672I DELALL command for *function* queued and released, key=*key*

Explanation: The system accepted the DELALL command for the function *function* and key *key*. These messages will be deleted.

User Response: None.

Module: IMRECMD

IMR673W DELALL command requires a key pattern

Explanation: The command DELALL requires a key pattern that defines the messages to be deleted.

User Response: Enter the command together with a key pattern.

Module: IMRECMD

IMR674I CLEAR command for *function* queued and released

Explanation: The system accepted the CLEAR command for the function *function*. All lists, retrievals, and responses produced by the SQL processor for the current user will be deleted.

User Response: None.

Module: IMRECMD

IMR681E DSLSRVP *request*, rc=*X'rc'*, rsn=*X'rsn'*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREI9X

IMR682E DSLTOFSV *request*, field=*fld*, rc=*X'rc'*, rsn=*X'rsn'*

Explanation: A *request* call to the serving module DSLTOFSV failed with return code *rc* and reason code *rsn* for field *fld*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREI9X

IMR683E DSLMMFS *request*, rc=*X'rc'*, rsn=*X'rsn'*

Explanation: A *request* call to the serving module DSLMMFS failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREI9X

IMR684E Command not allowed in this processing state

Explanation: The I9x imbed commands are accepted in an unprotected data entry function during display of a S.W.I.F.T. user-to-user message.

User Response: None.

Module: IMREI9X

IMR685E Command parameter must be X to exclude fields

Explanation: To exclude fields of the original message and to insert an empty F79, specify X as parameter for an imbed command.

User Response: None.

Module: IMREI9X

IMR686E No valid user-user SWIFT message

Explanation: Only S.W.I.F.T. user-to-user message can be embedded into N9X envelopes.

User Response: None.

Module: IMREI9X

IMR687E No SWIFT input message

Explanation: The N92 envelope is only for S.W.I.F.T. input messages.

User Response: None.

Module: IMREI9X

IMR688E No data entry function

Explanation: The embed commands I9x are accepted in an unprotected data entry function during display of a S.W.I.F.T. user-to-user message.

User Response: None.

Module: IMREI9X

IMR700A Enter a Traffic Reconciliation command (status=CONN)

Explanation: The Traffic Reconciliation SQL processor is running and ready for processing queries.

User Response: To stop the program enter either STOP or TERM at the system console. To disconnect the program enter DISC at the system console.

Module: IMRSQLP

IMR701A Enter a Traffic Reconciliation command (status=DISC)

Explanation: The Traffic Reconciliation SQL processor is running but disconnected from the MERVA system.

User Response: To connect the program, enter CONN at the system console. To stop it, enter either STOP or TERM at the system console.

Module: IMRSQLP

IMR702E DSLSRVP *request*, **module=***mod*, **rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: A *request* call for module *mod* failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLP

IMR703E DSLSRVP *request*, **rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLP

IMR704E DSLMMFS *request*, **rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: A *request* call to the serving module DSLMMFS failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLP

IMR706E DSLTOFSV *request*, **rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: A *request* call to the serving module DSLTOFSV failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLP

IMR707E DSLNICT *request*, **rc=***X'rc'*

Explanation: A *request* call for DSLNICT failed with return code *rc*. The code is given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLP

IMR708E DSLQMGT *request*, **rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: A *request* call for DSLQMGT failed with return code *rc*. *rsn* is the routing scanner reason code (DSLRTNSC). The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLP

IMR709E DSLTOFSV *request*, **field=***fld*, **rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: A *request* call to the serving module DSLTOFSV failed with return code *rc* and reason code *rsn* for field *fld*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLP

IMR710E Request failed, module=*mod*, **rc=***X'rc'*

Explanation: During processing of a request found in the input queue of the SQL processor, a severe error occurred. The responsible subprogram *mod* ended with return code *rc*.

The return code can be one of:

4 Suspend processing

8 Terminate

User Response: To get more information about the error, inspect the job protocol of the SQL processor, and the processing log in the query response.

Module: IMRSQLP

IMR711I Stop condition reached, SQL processor terminates

Explanation: The SQL processor attempted to connect to MERVA, then terminated. Parameter SRVSTOP in the parameters module IMRPRM determines how often IMRSQLP attempts to connect to MERVA before terminating.

User Response: None.

Module: IMRSQLP

IMR721I Invalid request put on error queue *queue*

Explanation: An invalid request was found in the input queue of the SQL processor. The subprogram for invalid requests put it on the error queue *queue*.

User Response: Inspect the request in the error queue to get the reason for rejecting the request.

Module: IMRSQLPI

IMR722E DSLSRVP *request*, **rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPI

IMR723E DSLNICT *request*, **rc=***X'rc'*

Explanation: A *request* call for DSLNICT failed with return code *rc*. The code is given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPI

IMR724E DSLQMGT *request*, **rc=***X'rc'*

Explanation: A *request* call for DSLQMGT failed with return code *rc*. The code is given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPI

IMR725E DSLMMFS *request*, **rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: A *request* call to the serving module DSLMMFS failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPI

IMR726E DSLTOFSV *request*, **rc=***X'rc'*, **rsn=***X'rsn'*, **field=***fld*

Explanation: A *request* call to the serving module DSLTOFSV failed with return code *rc* and reason code *rsn* for field *fld*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPI

IMR731I DELALL cmd: rc=*rc*, **queue=***function*, **count=***nnnn*, **user=***id*

Explanation: A DELALL request from user *id* completed with return *rc*. *nnnn* messages were deleted from the queue *function*.

The return codes are:

0 OK, or SQL error

4 A queue element was busy, or there were no messages to delete

8 The DELALL request was invalid

User Response: None.

Module: IMRSQLPD

IMR732E DSLSRVP *request*, **rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPD

IMR733E DSLNICT *request*, **rc=***X'rc'*

Explanation: A *request* call for DSLNICT failed with return code *rc*. The code is given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPD

IMR734E DSLQMGT *request*, **rc=***X'rc'*

Explanation: A *request* call for DSLQMGT failed with return code *rc*. The code is given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPD

IMR736E DSLTOFSV *request*, **rc=***X'rc'*, **rsn=***X'rsn'*, **field=***fld*

Explanation: A *request* call to the serving module DSLTOFSV failed with return code *rc* and reason code *rsn* for field *fld*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPD

IMR741I *query* **list: rc=***rc*, **count=***nnn*, **user=***id*, **skipped=***mmm*

Explanation: A *query* request for generating a list was found in the input queue of the SQL processor. The subprogram for list generation built a list with *nnn* entries and ended with return code *rc*.

The return codes are:

0 OK

4 SQL result table empty

6 MERVA not ready, or QDS full

8 SQL incorrect

12 Terminate; MERVA or SQL error

User Response: None.

Module: IMRSQLPL

IMR742E DSLSRVP *request*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPL

IMR743E DSLNICT *request*, **rc=***X′rc′*

Explanation: A *request* call for DSLNICT failed with return code *rc*. The code is given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPL

IMR744E DSLQMGT *request*, **rc=***X′rc′*

Explanation: A *request* call for DSLQMGT failed with return code *rc*. The code is given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPL

IMR745E DSLMMFS *request*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call to the serving module DSLMMFS failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPL

IMR746E DSLTOFSV *request*, **rc=***X′rc′*, **rsn=***X′rsn′*, **field=***fld*

Explanation: A *request* call to the serving module DSLTOFSV failed with return code *rc* and reason code *rsn* for field *fld*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPL

IMR748E DSLMMFS *request*, **rc=***X′rc′*, **rsn=***X′rsn′*, **mcb=***mcb*

Explanation: A *request* call using the MCB *mcb* to the serving module DSLMMFS failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPL

IMR749E SQL error, request=*req*, **SQLCODE=***rc*

Explanation: An SQL statement *req* failed with SQLCODE *rc*.

User Response: See *DB2 for OS/390 Messages and Codes* for an explanation.

Module: IMRSQLPL

IMR750E Unsupported SQL data type *nnn*

Explanation: A list query specified the SQL data type with SQLTYPE value *nnn*. This type is not supported.

User Response: Refer to SQLTYPE in *DB2 for OS/390 SQL Reference* for the possible values of *nnn*.

Module: IMRSQLPL

IMR751E Initialization of query=*query* **failed**

Explanation: An error occurred during preparation of the query.

User Response: See the processing log in the request and the job protocol of the SQL processor for further information.

Module: IMRSQLPL

IMR752E Empty query

Explanation: The SQL query string has a length of 0.

User Response: Re-initiate the query and use the NOPROMPT command to display the query before it is submitted. Check the MCB used to generate the query string.

Module: IMRSQLPL

IMR761I *query***: rc=***rc*, **count=***nnn*, **user=***id*, **skipped=***mmm*

Explanation: An MCB-based query *query* from user *id* for message retrieval was processed with return code *rc*. *mmm* messages in the result table were skipped, and then *nnn* messages were retrieved.

The return codes are:

0	OK
4	SQL result table empty
6	MERVA not ready, or QDS full
8	SQL incorrect
12	Terminate; MERVA or SQL error

User Response: None.

Module: IMRSQLPR

IMR762E DSLSRVP *request*, **rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPR

IMR763E DSLNICT *request*, **rc=***X'rc'*

Explanation: A *request* call for DSLNICT failed with return code *rc*. The code is given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPR

IMR764E DSLQMGT *request*, **rc=***X'rc'*

Explanation: A *request* call for DSLQMGT failed with return code *rc*. The code is given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPR

IMR765E DSLMMFS *request*, **rc=***X'rc'*, **rsn=***X'rsn'*

Explanation: A *request* call to the serving module DSLMMFS failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPR

IMR766E DSLTOFSV *request*, **rc=***X'rc'*, **rsn=***X'rsn'*, **field=***fld*

Explanation: A *request* call to the serving module DSLTOFSV failed with return code *rc* and reason code *rsn* for field *fld*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPR

IMR768E DSLMMFS *request*, **rc=***X'rc'*, **rsn=***X'rsn'*, **mcb=***mcb*

Explanation: A *request* call using the MCB *mcb* to the serving module DSLMMFS failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPR

IMR769E SQL error, request=*req*, **SQLCODE=***rc*

Explanation: An SQL statement *req* failed with SQLCODE *rc*.

User Response: See *DB2 for OS/390 Messages and Codes* for an explanation.

Module: IMRSQLPR

IMR770E Unsupported SQL data type *nnn*

Explanation: A retrieval query specified the SQL data type with SQLTYPE value *nnn*. This type is not supported.

User Response: Refer to SQLTYPE in *DB2 for OS/390 SQL Reference* for the possible values of *nnn*.

Module: IMRSQLPR

IMR771E Initialization of query=*query* **failed**

Explanation: An error occurred during preparation of the query.

User Response: See the processing log in the request and the job protocol of the SQL processor for further information.

Module: IMRSQLPR

IMR772E Empty query

Explanation: The SQL query string has a length of 0.

User Response: Re-initiate the query and use the NOPROMPT command to display the query before it is submitted. Check the MCB used to generate the query string.

Module: IMRSQLPR

IMR781E DSLNICT *request*, **rc=***X'rc'*

Explanation: A *request* call for DSLNICT failed with return code *rc*. The code is given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLC

IMR782W Query processing terminated due to QDS/LMC load

Explanation: During processing of a query the SQL processor detected an overload of the queue data set or the large message cluster. Processing of the query stopped.

User Response: Reduce the load by deleting messages.

Module: IMRSQLC

IMR791I **CLEAR cmd: rc=*rc*, queue=*func*, count=*nnn*, user=*id***

Explanation: A CLEAR request was processed for user *id* and ended with return code *rc*. *nnn* messages were deleted from queue *func*.

The return code can be:

0 OK

8 Terminate

User Response: None.

Module: IMRSQLPC

IMR792E **DSLSRVP** *request*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPC

IMR793E **DSLNICT** *request*, **rc=***X′rc′*

Explanation: A *request* call for DSLNICT failed with return code *rc*. The code is given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPC

IMR794E **DSLQMGT** *request*, **rc=***X′rc′*

Explanation: A *request* call for DSLQMGT failed with return code *rc*. The code is given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPC

IMR796E **DSLTOFSV** *request*, **rc=***X′rc′*, **rsn=***X′rsn′*, **field=***fld*

Explanation: A *request* call to the serving module DSLTOFSV failed with return code *rc* and reason code *rsn* for field *fld*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMRSQLPC

IMR901E **DSLTOFSV** *request*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call to the serving module DSLTOFSV failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREG9X

IMR902E **DSLTOFSV** *request*, **rc=***X′rc′*, **rsn=***X′rsn′*, **field=***fld*

Explanation: A *request* call to the serving module DSLTOFSV failed with return code *rc* and reason code *rsn* for field *fld*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREG9X

IMR903E **DSLSRVP** *request*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call to the service module DSLSRVP failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREG9X

IMR904W **Command allowed in message selection**

Explanation: Message can be retrieved only in message selection status.

User Response: Change to message selection and reissue the command.

Module: IMREG9X

IMR905E **DSLMMFS** *request*, **rc=***X′rc′*, **rsn=***X′rsn′*

Explanation: A *request* call to the serving module DSLMMFS failed with return code *rc* and reason code *rsn*. The codes are given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREG9X

IMR906W **Command allowed in unprotected data entry function**

Explanation: Message can only be retrieved in unprotected data entry functions.

User Response: Change the MERVA function to retrieve messages.

Module: IMREG9X

IMR907E **DSLNICT** *request*, **rc=***X′rc′*

Explanation: A *request* call for DSLNICT failed with return code *rc*. The code is given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREG9X

IMR908E DSLQMGT *request***, rc=***X'rc'*

Explanation: A *request* call for DSLQMGT failed with return code *rc*. The code is given in hexadecimal.

User Response: See *MERVA for ESA V4 Messages and Codes* for an explanation.

Module: IMREG9X

IMR909W First parameter must be I(SN) or O(SN)

Explanation: The first parameter of the G9X commands specifies whether the retrieval is a S.W.I.F.T. input or S.W.I.F.T. output message.

User Response: Specify I or O as first parameter.

Module: IMREG9X

IMR910W Correspondent destination must be 8 or 11 characters

Explanation: A S.W.I.F.T. destination must be 8 characters and 11 characters if it includes the branch code.

User Response: Correct the command input.

Module: IMREG9X

IMR911W No message found

Explanation: The command G9X for retrieving a message failed because no message was found matching the selection criteria or you are not allowed to access the message. The G9X commands retrieve only messages belonging to the user's destination specified in the MERVA user profile.

User Response: None.

Module: IMREG9X

IMR912W Specify TRN and correspondent to identify message

Explanation: The TRN does not clearly identify the message to be retrieved.

User Response: Reissue the command with the correspondent destination as third parameter.

Module: IMREG9X

IMR913E SQL request=*req***, SQLCODE=***rc*

Explanation: An SQL *req* statement failed with SQLCODE *rc*.

User Response: See *DB2 for OS/390 Messages and Codes* for an explanation.

Module: IMREG9X

IMR914W Invalid SWIFT message cannot be processed

Explanation: The S.W.I.F.T. message is corrupted and cannot be processed.

User Response: None.

Module: IMREG9X

IMR915W N92 must refer to a SWIFT input message

Explanation: The G92 command can only be used to retrieve S.W.I.F.T. input messages.

User Response: None.

Module: IMREG9X

IMR916I SWIFT *input|output* **message retrieved by** *key*

Explanation: A S.W.I.F.T. message is retrieved successfully by TRN, ISN, or OSN.

User Response: Select the message by QSN and process the message.

Module: IMREG9X

IMR917W Message could not be clearly identified

Explanation: There are several messages matching the selection criteria.

User Response: Use a Traffic Reconciliation query to retrieve all the messages matching the criteria and select the appropriate one.

Module: IMREG9X

IMR918E *text*

Explanation: *text* is an error message generated by DB2 DSNTIAR.

System Action: The program terminates.

User Response: Use these messages and the preceding message IMR913E to diagnose the DB2 error.

Refer to *DB2 for OS/390 Messages and Codes* for an explanation of these messages.

Module: IMREG9X

Appendix D. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Deutschland
Informationssysteme GmbH
Department 3982
Pascalstrasse 100

70569 Stuttgart
Germany

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement or any equivalent agreement between us.

The following paragraph does apply to the US only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

Programming Interface Information

General-Use Programming Interfaces allow the customer to write programs that obtain the services of MERVA.

However, this book also documents Product-Sensitive Programming Interface and Associated Guidance Information provided by MERVA.

Product-Sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of this IBM software product. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-Sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-Sensitive Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a chapter or section by the following marking:

Product-Sensitive Programming Interface

Product-Sensitive Programming Interface and Associated Guidance Information...

End of Product-Sensitive Programming Interface

Trademarks

The following terms are trademarks of the IBM Corporation in the United States, other countries, or both:

- AIX
- C/370
- CICS
- CICS/ESA
- CICS/MVS
- CICS/VSE
- DB2
- IBM
- IMS/ESA
- MQSeries
- MVS
- MVS/ESA
- MVS/XA
- OS/2
- OS/390
- QMF
- RACF
- S/390
- VSE/ESA
- VTAM

Workstation (AWS) and Directory Services Application (DSA) are trademarks of S.W.I.F.T., La Hulpe in Belgium.

Pentium is a trademark of Intel Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

C-bus is a trademark of Corollary, Inc.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Glossary of Terms and Abbreviations

This glossary defines terms as they are used in this book. If you do not find the terms you are looking for, refer to the *IBM Dictionary of Computing*, New York: McGraw-Hill, and the *S.W.I.F.T. User Handbook*.

A

ACB. Access method control block.

ACC. MERVA Link USS application control command application. It provides a means of operating MERVA Link USS in USS shell and MVS batch environments.

access method control block (ACB). A control block that links an application program to VSAM or VTAM.

ACD. MERVA Link USS application control daemon.

ACT. MERVA Link USS application control table.

address. See *S.W.I.F.T. address*.

address expansion. The process by which the full name of a financial institution is obtained using the S.W.I.F.T. address, telex correspondent's address, or a nickname.

AMPDU. Application message protocol data unit, which is defined in the MERVA Link P1 protocol, and consists of an envelope and its content.

answerback. In telex, the response from the dialed correspondent to the WHO R U signal.

answerback code. A group of up to 6 letters following or contained in the answerback. It is used to check the answerback.

APC. Application control.

API. Application programming interface.

APPC. Advanced Program-to-Program Communication based on SNA LU 6.2 protocols.

APPL. A VTAM definition statement used to define a VTAM application program.

application programming interface (API). An interface that programs can use to exchange data.

application support filter (ASF). In MERVA Link, a user-written program that can control and modify any data exchanged between the Application Support Layer and the Message Transfer Layer.

application support process (ASP). An executing instance of an application support program. Each application support process is associated with an ASP entry in the partner table. An ASP that handles outgoing messages is a *sending ASP*; one that handles incoming messages is a *receiving ASP*.

application support program (ASP). In MERVA Link, a program that exchanges messages and reports with a specific remote partner ASP. These two programs must agree on which conversation protocol they are to use.

ASCII. American Standard Code for Information Interchange. The standard code, using a coded set consisting of 7-bit coded characters (8 bits including parity check), used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

ASF. Application support filter.

ASF. (1) Application support process. (2) Application support program.

ASPDU. Application support protocol data unit, which is defined in the MERVA Link P2 protocol.

authentication. The S.W.I.F.T. security check used to ensure that a message has not changed during transmission, and that it was sent by an authorized sender.

authenticator key. A set of alphanumeric characters used for the authentication of a message sent via the S.W.I.F.T. network.

authenticator-key file. The file that stores the keys used during the authentication of a message. The file contains a record for each of your financial institution's correspondents.

B

Back-to-Back (BTB). A MERVA Link function that enables ASPs to exchange messages in the local MERVA Link node without using data communication services.

bank identifier code. A 12-character code used to identify a bank within the S.W.I.F.T. network. Also called a S.W.I.F.T. address. The code consists of the following subcodes:
- The bank code (4 characters)
- The ISO country code (2 characters)
- The location code (2 characters)
- The address extension (1 character)

- The branch code (3 characters) for a S.W.I.F.T. user institution, or the letters “BIC” for institutions that are not S.W.I.F.T. users.

Basic Security Manager (BSM). A component of VSE/ESA Version 2.4 that is invoked by the System Authorization Facility, and used to ensure signon and transaction security.

BIC. Bank identifier code.

BIC Bankfile. A tape of bank identifier codes supplied by S.W.I.F.T.

BIC Database Plus Tape. A tape of financial institutions and currency codes, supplied by S.W.I.F.T. The information is compiled from various sources and includes national, international, and cross-border identifiers.

BIC Directory Update Tape. A tape of bank identifier codes and currency codes, supplied by S.W.I.F.T., with extended information as published in the printed BIC Directory.

body. The second part of an IM-ASPDU. It contains the actual application data or the message text that the IM-AMPDU transfers.

BSC. Binary synchronous control.

BSM. Basic Security Manager.

BTB. Back-to-back.

buffer. A storage area used by MERVA programs to store a message in its internal format. A buffer has an 8-byte prefix that indicates its length.

C

CBT. S.W.I.F.T. computer-based terminal.

CCSID. Coded character set identifier.

CDS. Control data set.

central service. In MERVA, a service that uses resources that either require serialization of access, or are only available in the MERVA nucleus.

CF message. Confirmed message. When a sending MERVA Link system is informed of the successful delivery of a message to the receiving application, it routes the delivered application messages as CF messages, that is, messages of class CF, to an ACK wait queue or to a complete message queue.

COA. Confirm on arrival.

COD. Confirm on delivery.

coded character set identifier (CCSID). The name of a coded set of characters and their code point assignments.

commit. In MQSeries, to commit operations is to make the changes on MQSeries queues permanent. After putting one or more messages to a queue, a commit makes them visible to other programs. After getting one or more messages from a queue, a commit permanently deletes them from the queue.

confirm-on-arrival (COA) report. An MQSeries report message type created when a message is placed on that queue. It is created by the queue manager that owns the destination queue.

confirm-on-delivery (COD) report. An MQSeries report message type created when an application retrieves a message from the queue in a way that causes the message to be deleted from the queue. It is created by the queue manager.

control fields. In MERVA Link, fields that are part of a MERVA message on the queue data set and of the message in the TOF. Control fields are written to the TOF at nesting identifier 0. Messages in S.W.I.F.T. format do not contain control fields.

correspondent. An institution to which your institution sends and from which it receives messages.

correspondent identifier. The 11-character identifier of the receiver of a telex message. Used as a key to retrieve information from the Telex correspondents file.

cross-system coupling facility. See *XCF*.

coupling services. In a sysplex, the functions of XCF that transfer data and status information among the members of a group that reside in one or more of the MVS systems in the sysplex.

couple data set. See *XCF couple data set*.

CTP. MERVA Link command transfer processor.

currency code file. A file containing the currency codes, together with the name, fraction length, country code, and country names.

D

daemon. A long-lived process that runs unattended to perform continuous or periodic systemwide functions.

DASD. Direct access storage device.

data area. An area of a predefined length and format on a panel in which data can be entered or displayed. A field can consist of one or more data areas.

data element. A unit of data that, in a certain context, is considered indivisible. In MERVA Link, a data element consists of a 2-byte data element length field, a 2-byte data-element identifier field, and a field of variable length containing the data element data.

datagram. In TCP/IP, the basic unit of information passed across the Internet environment. This type of message does not require a reply, and is the simplest type of message that MQSeries supports.

data terminal equipment. That part of a data station that serves as a data source, data link, or both, and provides for the data communication control function according to protocols.

DB2. A family of IBM licensed programs for relational database management.

dead-letter queue. A queue to which a queue manager or application sends messages that it cannot deliver. Also called *undelivered-message queue*.

dial-up number. A series of digits required to establish a connection with a remote correspondent via the public telex network.

direct service. In MERVA, a service that uses resources that are always available and that can be used by several requesters at the same time.

display mode. The mode (PROMPT or NOPROMPT) in which S.W.I.F.T. messages are displayed. See *PROMPT mode* and *NOPROMPT mode.*

distributed queue management (DQM). In MQSeries message queuing, the setup and control of message channels to queue managers on other systems.

DQM. Distributed queue management.

DTE. Data terminal equipment.

E

EBCDIC. Extended Binary Coded Decimal Interchange Code. A coded character set consisting of 8-bit coded characters.

ECB. Event control block.

EDIFACT. Electronic Data Interchange for Administration, Commerce and Transport (a United Nations standard).

ESM. External security manager.

EUD. End-user driver.

exception report. An MQSeries report message type that is created by a message channel agent when a message is sent to another queue manager, but that message cannot be delivered to the specified destination queue.

external line format (ELF) messages. Messages that are not fully tokenized, but are stored in a single field in the TOF. Storing messages in ELF improves performance, because no mapping is needed, and checking is not performed.

external security manager (ESM). A security product that is invoked by the System Authorization Facility. RACF is an example of an ESM.

F

FDT. Field definition table.

field. In MERVA, a portion of a message used to enter or display a particular type of data in a predefined format. A field is located by its position in a message and by its tag. A field is made up of one or more data areas. See also *data area*.

field definition table (FDT). The field definition table describes the characteristics of a field; for example, its length and number of its data areas, and whether it is mandatory. If the characteristics of a field change depending on its use in a particular message, the definition of the field in the FDT can be overridden by the MCB specifications.

field group. One or several fields that are defined as being a group. Because a field can occur more than once in a message, field groups are used to distinguish them. A name can be assigned to the field group during message definition.

field group number. In the TOF, a number is assigned to each field group in a message in ascending order from 1 to 255. A particular field group can be accessed using its field group number.

field tag. A character string used by MERVA to identify a field in a network buffer. For example, for S.W.I.F.T. field 30, the field tag is **:30:**.

FIN. Financial application.

FIN-Copy. The MERVA component used for S.W.I.F.T. FIN-Copy support.

finite state machine. The theoretical base describing the rules of a service request's state and the conditions to state transitions.

FMT/ESA. MERVA-to-MERVA Financial Message Transfer/ESA.

form. A partially-filled message containing data that can be copied for a new message of the same message type.

G

GPA. General purpose application.

H

HFS. Hierarchical file system.

hierarchical file system (HFS). A system for organizing files in a hierarchy, as in a UNIX system. OS/390 UNIX System Services files are organized in an HFS. All files are members of a directory, and each directory is in turn a member of a directory at a higher level in the HFS. The highest level in the hierarchy is the root directory.

I

IAM. Interapplication messaging (a MERVA Link message exchange protocol).

IM-ASPDU. Interapplication messaging application support protocol data unit. It contains an application message and consists of a heading and a body.

incore request queue. Another name for the request queue to emphasize that the request queue is held in memory instead of on a DASD.

InetD. Internet Daemon. It provides TCP/IP communication services in the OS/390 USS environment.

initiation queue. In MQSeries, a local queue on which the queue manager puts trigger messages.

input message. A message that is input into the S.W.I.F.T. network. An input message has an input header.

INTERCOPE TelexBox. This telex box supports various national conventions for telex procedures and protocols.

interservice communication. In MERVA ESA, a facility that enables communication among services if MERVA ESA is running in a multisystem environment.

intertask communication. A facility that enables application programs to communicate with the MERVA nucleus and so request a central service.

IP. Internet Protocol.

IP message. In-process message. A message that is in the process of being transferred to another application.

ISC. Intersystem communication.

ISN. Input sequence number.

ISN acknowledgment. A collective term for the various kinds of acknowledgments sent by the S.W.I.F.T. network.

ISO. International Organization for Standardization.

ITC. Intertask communication.

J

JCL. Job control language.

journal. A chronological list of records detailing MERVA actions.

journal key. A key used to identify a record in the journal.

journal service. A MERVA central service that maintains the journal.

K

KB. Kilobyte (1024 bytes).

key. A character or set of characters used to identify an item or group of items. For example, the user ID is the key to identify a user file record.

key-sequenced data set (KSDS). A VSAM data set whose records are loaded in key sequence and controlled by an index.

keyword parameter. A parameter that consists of a keyword, followed by one or more values.

KSDS. Key-sequenced data set.

L

LAK. Login acknowledgment message. This message informs you that you have successfully logged in to the S.W.I.F.T. network.

large message. A message that is stored in the large message cluster (LMC). The maximum length of a message to be stored in the VSAM QDS is 31900 bytes. Messages up to 2 MB can be stored in the LMC. For queue management using DB2 no distinction is made between messages and large messages.

large queue element. A queue element that is larger than the smaller of:

- The limiting value specified during the customization of MERVA
- 32 KB

LC message. Last confirmed control message. It contains the message-sequence number of the application or acknowledgment message that was last confirmed; that is, for which the sending MERVA Link system most recently received confirmation of a successful delivery.

LDS. Logical data stream.

LMC. Large message cluster.

LNK. Login negative acknowledgment message. This message indicates that the login to the S.W.I.F.T. network has failed.

local queue. In MQSeries, a queue that belongs to a local queue manager. A local queue can contain a list of messages waiting to be processed. Contrast with *remote queue*.

local queue manager. In MQSeries, the queue manager to which the program is connected, and that provides message queuing services to that program. Queue managers to which a program is not connected are remote queue managers, even if they are running on the same system as the program.

login. To start the connection to the S.W.I.F.T. network.

LR message. Last received control message, which contains the message-sequence number of the application or acknowledgment message that was last received from the partner application.

LSN. Login sequence number.

LT. See *LTERM*.

LTC. Logical terminal control.

LTERM. Logical terminal. Logical terminal names have 4 characters in CICS and up to 8 characters in IMS.

LU. A VTAM logical unit.

M

maintain system history program (MSHP). A program used for automating and controlling various installation, tailoring, and service activities for a VSE system.

MCA. Message channel agent.

MCB. Message control block.

MERVA ESA. The IBM licensed program Message Entry and Routing with Interfaces to Various Applications for ESA.

MERVA Link. A MERVA component that can be used to interconnect several MERVA systems.

message. A string of fields in a predefined form used to provide or request information. See also *S.W.I.F.T. financial message.*

message channel. In MQSeries distributed message queuing, a mechanism for moving messages from one queue manager to another. A message channel comprises two message channel agents (a sender and a receiver) and a communication link.

message channel agent (MCA). In MQSeries, a program that transmits prepared messages from a transmission queue to a communication link, or from a communication link to a destination queue.

message control block (MCB). The definition of a message, screen panel, net format, or printer layout made during customization of MERVA.

Message Format Service (MFS). A MERVA direct service that formats a message according to the medium to be used, and checks it for formal correctness.

Message Integrity Protocol (MIP). In MERVA Link, the protocol that controls the exchange of messages between partner ASPs. This protocol ensures that any loss of a message is detected and reported, and that no message is duplicated despite system failures at any point during the transfer process.

message-processing function. The various parts of MERVA used to handle a step in the message-processing route, together with any necessary equipment.

message queue. See *queue*.

Message Queue Interface (MQI). The programming interface provided by the MQSeries queue managers. It provides a set of calls that let application programs access message queuing services such as sending messages, receiving messages, and manipulating MQSeries objects.

Message Queue Manager (MQM). An IBM licensed program that provides message queuing services. It is part of the MQSeries set of products.

message type (MT). A number, up to 7 digits long, that identifies a message. S.W.I.F.T. messages are identified by a 3-digit number; for example S.W.I.F.T. message type MT S100.

MFS. Message Format Service.

MIP. Message Integrity Protocol.

MPDU. Message protocol data unit, which is defined in P1.

MPP. In IMS, message-processing program.

MQH. MQSeries queue handler.

MQI. Message queue interface.

MQM. Message queue manager.

MQS. MQSeries nucleus server.

MQSeries. A family of IBM licensed programs that provides message queuing services.

MQSeries nucleus server (MQS). A MERVA component that listens for messages on an MQI queue, receives them, extracts a service request, and passes it via the request queue handler to another MERVA ESA instance for processing.

MQSeries queue handler (MQH). A MERVA component that performs service calls to the Message Queue Manager via the provided Message Queue Interface.

MSC. MERVA system control facility.

MSHP. Maintain system history program.

MSN. Message sequence number.

MT. Message type.

MTP. (1) Message transfer program. (2) Message transfer process.

MTS. Message Transfer System.

MTSP. Message Transfer Service Processor.

MTT. Message type table.

multisystem application. (1) An application program that has various functions distributed across MVS systems in a multisystem environment. (2) In XCF, an authorized application that uses XCF coupling services. (3) In MERVA ESA, multiple instances of MERVA ESA that are distributed among different MVS systems in a multisystem environment.

multisystem environment. An environment in which two or more MVS systems reside on one or more processors, and programs on one system can communicate with programs on the other systems. With XCF, the environment in which XCF services are available in a defined sysplex.

multisystem sysplex. A sysplex in which one or more MVS systems can be initialized as part of the sysplex. In a multisystem sysplex, XCF provides coupling services on all systems in the sysplex and requires an XCF couple data set that is shared by all systems. See also *single-system sysplex*.

MVS/ESA. Multiple Virtual Storage/Enterprise Systems Architecture.

N

namelist. An MQSeries for MVS/ESA object that contains a list of queue names.

nested message. A message that is composed of one or more message types.

nested message type. A message type that is contained in another message type. In some cases, only part of a message type (for example, only the mandatory fields) is nested, but this "partial" nested message type is also considered to be nested. For example, S.W.I.F.T. MT 195 could be used to request information about a S.W.I.F.T. MT 100 (customer transfer). The S.W.I.F.T. MT 100 (or at least its mandatory fields) is then nested in S.W.I.F.T. MT 195.

nesting identifier. An identifier (a number from 2 to 255) that is used to access a nested message type.

network identifier. A single character that is placed before a message type to indicate which network is to be used to send the message; for example, **S** for S.W.I.F.T.

network service access point (NSAP). The endpoint of a network connection used by the S.W.I.F.T. transport layer.

NOPROMPT mode. One of two ways to display a message panel. NOPROMPT mode is only intended for experienced SWIFT Link users who are familiar with the structure of S.W.I.F.T. messages. With NOPROMPT mode, only the S.W.I.F.T. header, trailer, and pre-filled fields and their tags are displayed. Contrast with *PROMPT mode*.

NSAP. Network service access point.

nucleus server. A MERVA component that processes a service request as selected by the request queue handler. The service a nucleus server provides and the way it provides it is defined in the nucleus server table (DSLNSVT).

O

object. In MQSeries, objects define the properties of queue managers, queues, process definitions, and namelists.

occurrence. See *repeatable sequence*.

option. One or more characters added to a S.W.I.F.T. field number to distinguish among different layouts for and meanings of the same field. For example, S.W.I.F.T. field 60 can have an option F to identify a first opening balance, or M for an intermediate opening balance.

origin identifier (origin ID). A 34-byte field of the MERVA user file record. It indicates, in a MERVA and SWIFT Link installation that is shared by several banks, to which of these banks the user belongs. This lets the user work for that bank only.

OSN. Output sequence number.

OSN acknowledgment. A collective term for the various kinds of acknowledgments sent to the S.W.I.F.T. network.

output message. A message that has been received from the S.W.I.F.T. network. An output message has an output header.

P

P1. In MERVA Link, a peer-to-peer protocol used by cooperating message transfer processes (MTPs).

P2. In MERVA Link, a peer-to-peer protocol used by cooperating application support processes (ASPs).

P3. In MERVA Link, a peer-to-peer protocol used by cooperating command transfer processors (CTPs).

packet switched public data network (PSPDN). A public data network established and operated by network common carriers or telecommunication administrations for providing packet-switched data transmission.

panel. A formatted display on a display terminal. Each page of a message is displayed on a separate panel.

parallel processing. The simultaneous processing of units of work by several servers. The units of work can be either transactions or subdivisions of larger units of work.

parallel sysplex. A sysplex that uses one or more coupling facilities.

partner table (PT). In MERVA Link, the table that defines how messages are processed. It consists of a header and different entries, such as entries to specify the message-processing parameters of an ASP or MTP.

PCT. Program Control Table (of CICS).

PDE. Possible duplicate emission.

PDU. Protocol data unit.

PF key. Program-function key.

positional parameter. A parameter that must appear in a specified location relative to other parameters.

PREMIUM. The MERVA component used for S.W.I.F.T. PREMIUM support.

process definition object. An MQSeries object that contains the definition of an MQSeries application. A queue manager uses the definitions contained in a process definition object when it works with trigger messages.

program-function key. A key on a display terminal keyboard to which a function (for example, a command) can be assigned. This lets you execute the function (enter the command) with a single keystroke.

PROMPT mode. One of two ways to display a message panel. PROMPT mode is intended for SWIFT Link users who are unfamiliar with the structure of S.W.I.F.T. messages. With PROMPT mode, all the fields and tags are displayed for the S.W.I.F.T. message. Contrast with *NOPROMPT mode*.

protocol data unit (PDU). In MERVA Link a PDU consists of a structured sequence of implicit and explicit data elements:

- Implicit data elements contain other data elements.
- Explicit data elements cannot contain any other data elements.

PSN. Public switched network.

PSPDN. Packet switched public data network.

PSTN. Public switched telephone network.

PT. Partner table.

PTT. A national post and telecommunication authority (post, telegraph, telephone).

Q

QDS. Queue data set.

QSN. Queue sequence number.

queue. (1) In MERVA, a logical subdivision of the MERVA queue data set used to store the messages associated with a MERVA message-processing function. A queue has the same name as the message-processing function with which it is associated. (2) In MQSeries, an object onto which message queuing applications can put messages, and from which they can get messages. A queue is owned and maintained by a queue manager. See also *request queue*.

queue element. A message and its related control information stored in a data record in the MERVA ESA Queue Data Set.

queue management. A MERVA service function that handles the storing of messages in, and the retrieval of messages from, the queues of message-processing functions.

queue manager. (1) An MQSeries system program that provides queuing services to applications. It provides an application programming interface so that programs can access messages on the queues that the queue manager owns. See also *local queue manager* and *remote queue manager*. (2) The MQSeries object that defines the attributes of a particular queue manager.

queue sequence number (QSN). A sequence number that is assigned to the messages stored in a logical queue by MERVA ESA queue management in ascending order. The QSN is always unique in a queue.

It is reset to zero when the queue data set is formatted, or when a queue management restart is carried out and the queue is empty.

R

RACF. Resource Access Control Facility.

RBA. Relative byte address.

RC message. Recovered message; that is, an IP message that was copied from the control queue of an inoperable or closed ASP via the **recover** command.

ready queue. A MERVA queue used by SWIFT Link to collect S.W.I.F.T. messages that are ready for sending to the S.W.I.F.T. network.

remote queue. In MQSeries, a queue that belongs to a remote queue manager. Programs can put messages on remote queues, but they cannot get messages from remote queues. Contrast with *local queue*.

remote queue manager. In MQSeries, a queue manager is remote to a program if it is not the queue manager to which the program is connected.

repeatable sequence. A field or a group of fields that is contained more than once in a message. For example, if the S.W.I.F.T. fields 20, 32, and 72 form a sequence, and if this sequence can be repeated up to 10 times in a message, each sequence of the fields 20, 32, and 72 would be an occurrence of the repeatable sequence.

In the TOF, the occurrences of a repeatable sequence are numbered in ascending order from 1 to 32767 and can be referred to using the occurrence number.

A repeatable sequence in a message may itself contain another repeatable sequence. To identify an occurrence within such a nested repeatable sequence, more than one occurrence number is necessary.

reply message. In MQSeries, a type of message used for replies to request messages.

reply-to queue. In MQSeries, the name of a queue to which the program that issued an MQPUT call wants a reply message or report message sent.

report message. In MQSeries, a type of message that gives information about another message. A report message usually indicates that the original message cannot be processed for some reason.

request message. In MQSeries, a type of message used for requesting a reply from another program.

request queue. The queue in which a service request is stored. It resides in main storage and consists of a set of request queue elements that are chained in different queues:

- Requests waiting to be processed
- Requests currently being processed
- Requests for which processing has finished

request queue handler (RQH). A MERVA ESA component that handles the queueing and scheduling of service requests. It controls the request processing of a nucleus server according to rules defined in the finite state machine.

Resource Access Control Facility (RACF). An IBM licensed program that provides for access control by identifying and verifying users to the system, authorizing access to protected resources, logging detected unauthorized attempts to enter the system, and logging detected accesses to protected resources.

retype verification. See *verification*.

routing. In MERVA, the passing of messages from one stage in a predefined processing path to the next stage.

RP. Regional processor.

RQH. Request queue handler.

RRDS. Relative record data set.

S

SAF. System Authorization Facility.

SCS. SNA character string.

SCP. System control process.

SDI. Sequential data set input. A batch utility used to import messages from a sequential data set or a tape into MERVA ESA queues.

SDO. Sequential data set output. A batch utility used to export messages from a MERVA ESA queue to a sequential data set or a tape.

SDY. Sequential data set system printer. A batch utility used to print messages from a MERVA ESA queue.

service request. A type of request that is created and passed to the request queue handler whenever a nucleus server requires a service that is not currently available.

sequence number. A number assigned to each message exchanged between two nodes. The number is increased by one for each successive message. It starts from zero each time a new session is established.

sign off. To end a session with MERVA.

sign on. To start a session with MERVA.

single-system sysplex. A sysplex in which only one MVS system can be initialized as part of the sysplex. In a single-system sysplex, XCF provides XCF services on the system, but does not provide signaling services between MVS systems. A single-system sysplex requires an XCF couple data set. See also *multisystem sysplex*.

small queue element. A queue element that is smaller than the smaller of:

- The limiting value specified during the customization of MERVA
- 32 KB

SMP/E. System Modification Program Extended.

SN. Session number.

SNA. Systems Network Architecture.

SNA character string. In SNA, a character string composed of EBCDIC controls, optionally mixed with user data, that is carried within a request or response unit.

SPA. Scratch pad area.

SQL. Structured Query Language.

SR-ASPDU. The status report application support PDU, which is used by MERVA Link for acknowledgment messages.

SSN. Select sequence number.

subfield. A subdivision of a field with a specific meaning. For example, the S.W.I.F.T. field 32 has the subfields date, currency code, and amount. A field can have several subfield layouts depending on the way the field is used in a particular message.

SVC. (1) Switched Virtual Circuit. (2) Supervisor call instruction.

S.W.I.F.T. (1) Society for Worldwide Interbank Financial Telecommunication s.c. (2) The network provided and managed by the Society for Worldwide Interbank Financial Telecommunication s.c.

S.W.I.F.T. address. Synonym for *bank identifier code*.

S.W.I.F.T. Correspondents File. The file containing the bank identifier code (BIC), together with the name, postal address, and zip code of each financial institution in the BIC Directory.

S.W.I.F.T. financial message. A message in one of the S.W.I.F.T. categories 1 to 9 that you can send or receive via the S.W.I.F.T. network. See *S.W.I.F.T. input message* and *S.W.I.F.T. output message*.

S.W.I.F.T. header. The leading part of a message that contains the sender and receiver of the message, the message priority, and the type of message.

S.W.I.F.T. input message. A S.W.I.F.T. message with an input header to be sent to the S.W.I.F.T. network.

SWIFT link. The MERVA ESA component used to link to the S.W.I.F.T. network.

S.W.I.F.T. network. Refers to the S.W.I.F.T. network of the Society for Worldwide Interbank Financial Telecommunication (S.W.I.F.T.).

S.W.I.F.T. output message. A S.W.I.F.T. message with an output header coming from the S.W.I.F.T. network.

S.W.I.F.T. system message. A S.W.I.F.T. general purpose application (GPA) message or a financial application (FIN) message in S.W.I.F.T. category 0.

switched virtual circuit (SVC). An X.25 circuit that is dynamically established when needed. It is the X.25 equivalent of a switched line.

sysplex. One or more MVS systems that communicate and cooperate via special multisystem hardware components and software services.

System Authorization Facility (SAF). An MVS or VSE facility through which MERVA ESA communicates with an external security manager such as RACF (for MVS) or the basic security manager (for VSE).

System Control Process (SCP). A MERVA Link component that handles the transfer of MERVA ESA commands to a partner MERVA ESA system, and the receipt of the command response. It is associated with a system control process entry in the partner table.

System Modification Program Extended (SMP/E). A licensed program used to install software and software changes on MVS systems.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operating sequences for transmitting information units through, and for controlling the configuration and operation of, networks.

T

tag. A field identifier.

TCP/IP. Transmission Control Protocol/Internet Protocol.

Telex Correspondents File. A file that stores data about correspondents. When the user enters the corresponding nickname in a Telex message, the corresponding information in this file is automatically retrieved and entered into the Telex header area.

telex header area. The first part of the telex message. It contains control information for the telex network.

telex interface program (TXIP). A program that runs on a Telex front-end computer and provides a communication facility to connect MERVA ESA with the Telex network.

Telex Link. The MERVA ESA component used to link to the public telex network via a Telex substation.

Telex substation. A unit comprised of the following:

- Telex Interface Program
- A Telex front-end computer
- A Telex box

Terminal User Control Block (TUCB). A control block containing terminal-specific and user-specific information used for processing messages for display devices such as screen and printers.

test key. A key added to a telex message to ensure message integrity and authorized delivery. The test key is an integer value of up to 16 digits, calculated manually or by a test-key processing program using the significant information in the message, such as amounts, currency codes, and the message date.

test-key processing program. A program that automatically calculates and verifies a test key. The Telex Link supports panels for input of test-key-related data and an interface for a test-key processing program.

TFD. Terminal feature definitions table.

TID. Terminal identification. The first 9 characters of a bank identifier code (BIC).

TOF. Originally the abbreviation of *tokenized form*, the TOF is a storage area where messages are stored so that their fields can be accessed directly by their field names and other index information.

TP. Transaction program.

transaction. A specific set of input data that triggers the running of a specific process or job; for example, a message destined for an application program.

transaction code. In IMS and CICS, an alphanumeric code that calls an IMS message processing program or a CICS transaction. Transaction codes have 4 characters in CICS and up to 8 characters in IMS.

Transmission Control Protocol/Internet Protocol (TCP/IP). A set of communication protocols that support peer-to-peer connectivity functions for both local and wide area networks.

transmission queue. In MQSeries, a local queue on which prepared messages destined for a remote queue manager are temporarily stored.

trigger event. In MQSeries, an event (such as a message arriving on a queue) that causes a queue manager to create a trigger message on an initiation queue.

trigger message. In MQSeries, a message that contains information about the program that a trigger monitor is to start.

trigger monitor. In MQSeries, a continuously-running application that serves one or more initiation queues. When a trigger message arrives on an initiation queue, the trigger monitor retrieves the message. It uses the information in the trigger message to start a process that serves the queue on which a trigger event occurred.

triggering. In MQSeries, a facility that allows a queue manager to start an application automatically when predetermined conditions are satisfied.

TUCB. Terminal User Control Block.

TXIP. Telex interface program.

U

UMR. Unique message reference.

unique message reference (UMR). An optional feature of MERVA ESA that provides each message with a unique identifier the first time it is placed in a queue. It is composed of a MERVA ESA installation name, a sequence number, and a date and time stamp.

UNIT. A group of related literals or fields of an MCB definition, or both, enclosed by a DSLLUNIT and DSLLUEND macroinstruction.

UNIX System Services (USS). A component of OS/390, formerly called OpenEdition (OE), that creates a UNIX environment that conforms to the XPG4 UNIX 1995 specifications, and provides two open system interfaces on the OS/390 operating system:

- An application program interface (API)
- An interactive shell interface

UN/EDIFACT. United Nations Standard for Electronic Data Interchange for Administration, Commerce, and Transport.

USE. S.W.I.F.T. User Security Enhancements.

user file. A file containing information about all MERVA ESA users; for example, which functions each user is allowed to access. The user file is encrypted and can only be accessed by authorized persons.

user identification and verification. The acts of identifying and verifying a RACF-defined user to the system during logon or batch job processing. RACF identifies the user by the user ID and verifies the user by the password or operator identification card supplied during logon processing or the password supplied on a batch JOB statement.

USS. UNIX System Services.

V

verification. Checking to ensure that the contents of a message are correct. Two kinds of verification are:

- Visual verification, in which you read the message and confirm that you have done so
- Retype verification, in which you reenter the data to be verified

Virtual LU. An LU defined in MERVA Extended Connectivity for communication between MERVA and MERVA Extended Connectivity.

Virtual Storage Access Method (VSAM). An access method for direct or sequential processing of fixed and variable-length records on direct access devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry sequence), or by relative-record number.

Virtual Telecommunications Access Method (VTAM). An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability.

VSAM. Virtual Storage Access Method.

VTAM. Virtual Telecommunications Access Method (IBM licensed program).

X

X.25. An ISO standard for interface to packet switched communications services.

XCF. Abbreviation for *cross-system coupling facility*, which is a special logical partition that provides high-speed caching, list processing, and locking functions in a sysplex. XCF provides the MVS coupling services that allow authorized programs on MVS systems in a multisystem environment to communicate with (send data to and receive data from) authorized programs on other MVS systems.

XCF couple data set. A data set that is created through the XCF couple data set format utility and, depending on its designated type, is shared by some or all of the MVS systems in a sysplex. It is accessed only by XCF and contains XCF-related data about the sysplex, systems, applications, groups, and members.

XCF group. The set of related members defined to SCF by a multisystem application in which members of the group can communicate with (send data to and receive data from) other members of the same group. All MERVA systems working together in a sysplex must pertain to the same XCF group.

XCF member. A specific function of a multisystem application that is defined to XCF and assigned to a group by the multisystem application. A member resides on one system in a sysplex and can use XCF services to communicate with other members of the same group.

Bibliography

MERVA ESA Publications

- *MERVA for ESA Version 4 Application Programming Interface Guide*, SH12-6374
- *MERVA for ESA Version 4 Advanced MERVA Link*, SH12-6390
- *MERVA for ESA Version 4 Concepts and Components*, SH12-6381
- *MERVA for ESA Version 4 Customization Guide*, SH12-6380
- *MERVA for ESA Version 4 Diagnosis Guide*, SH12-6382
- *MERVA for ESA Version 4 Installation Guide*, SH12-6378
- *MERVA for ESA Version 4 Licensed Program Specifications*, GH12-6373
- *MERVA for ESA Version 4 Macro Reference*, SH12-6377
- *MERVA for ESA Version 4 Messages and Codes*, SH12-6379
- *MERVA for ESA Version 4 Operations Guide*, SH12-6375
- *MERVA for ESA Version 4 System Programming Guide*, SH12-6366
- *MERVA for ESA Version 4 User's Guide*, SH12-6376

Other MERVA Publications

- *MERVA Directory Services*, SH12-6367
- *MERVA Extended Connectivity Installation and User's Guide*, SH12-6157
- *MERVA Extended Connectivity Licensed Program Specifications*, GH12-6186
- *MERVA Message Processing Client for Windows NT User's Guide*, SH12-6341
- *MERVA Traffic Reconciliation*, SH12-6392
- *MERVA USE Administration Guide*, SH12-6338
- *MERVA USE & Branch for Windows NT Installation and Customization Guide*, SH12-6335
- *MERVA Workstation Based Functions*, SH12-6383

Other IBM Publications

- *CICS/ESA Version 4.1 Application Programming Reference*, SC33-1170
- *DB2 for OS/390 Messages and Codes*, GC26-8979
- *DB2 for OS/390 SQL Reference*, GC26-8979
- *DFSORT Application Programming Guide*, SC33-4035
- *DFSMS/MVS Macro Instructions for Data Sets*, SC26-4913
- *IMS/ESA Version 5 Application Programming: Transaction Manager*, SC26-8017
- *OS/390 MVS Assembler Services Reference*, GC28-1910
- *Query Management Facility Managing QMF for MVS*, SC26-8218
- *Query Management Facility User Guide*, SC26-8078

S.W.I.F.T. Publications

The following are published by the Society for Worldwide Interbank Financial Telecommunication, s.c., in La Hulpe, Belgium:

- *S.W.I.F.T. User Handbook*
- *S.W.I.F.T. Dictionary*
- *S.W.I.F.T. FIN Security Guide*
- *S.W.I.F.T. Card Readers User Guide*

Index

MERVA Requirement Request

Use the form overleaf to send us requirement requests for the MERVA product. Fill in the blank lines with the information that we need to evaluate and implement your request. Provide also information about your hardware and software environments and about the MERVA release levels used in your environment.

Provide a detailed description of your requirement. If you are requesting a new function, describe in full what you want that function to do. If you are requesting that a function be changed, briefly describe how the function works currently, followed by how you are requesting that it should work.

If you are a customer, provide us with the appropriate contacts in your organization to discuss the proposal and possible implementation alternatives.

If you are an IBM employee, include at least the name of one customer who has this requirement. Add the name and telephone number of the appropriate contacts in the customer's organization to discuss the proposal and possible implementation alternatives. If possible, send this requirement online to MERVAREQ at SDFVM1.

For comments on this book, use the form provided at the back of this publication.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Send the fax to:

To: MERVA Development, Dept. 5640
Attention: Gerhard Stubbe

IBM Deutschland Entwicklung GmbH
Schoenaicher Str. 220
D-71032 Boeblingen
Germany

Fax Number: +49-7031-16-4881
Internet address:
mervareq@de.ibm.com

MERVA Requirement Request

```
To: MERVA Development, Dept. 5640            Fax Number: +49-7031-16-4881
    Attention: Gerhard Strubbe               Internet address:
                                             mervareq@de.ibm.com
    IBM Deutschland Entwicklung GmbH
    Schoenaicher Str. 220
    D-71032 Boeblingen      Germany
```

Page 1 of ______

Customer's Name ____

Customer's Address ____

____

____

Customer's
Telephone/Fax ____

Contact Person at ____
Customer's Location
Telephone/Fax ____

MERVA
Version/Release ____

Operating System ____
Sub-System
Version/Release ____

Hardware ____

Requirement
Description ____

____

____

____

____

____

____

Expected Benefits ____

____

____

Readers’ Comments — We’d Like to Hear from You

MERVA ESA Components
Traffic Reconciliation

Publication No. SH12-6392-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	□	□	□	□	□

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	□	□	□	□	□
Complete	□	□	□	□	□
Easy to find	□	□	□	□	□
Easy to understand	□	□	□	□	□
Well organized	□	□	□	□	□
Applicable to your tasks	□	□	□	□	□

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? □ Yes □ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Company or Organization

Phone No.

Address

Readers' Comments — We'd Like to Hear from You

SH12-6392-00

IBM®

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Deutschland Entwicklung GmbH
Information Development, Dept. 0446
Schoenaicher Strasse 220
71032 Boeblingen
Germany

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Program Number: 5648-B30

Printed in Denmark by IBM Danmark A/S

SH12-6392-00

Spine information:

IBM MERVA ESA Components Traffic Reconciliation

Version 4
Release 1