

MERVA ESA Components



# MERVA USE & Branch for Windows NT Diagnosis Guide

*Version 4 Release 1*



MERVA ESA Components



# MERVA USE & Branch for Windows NT Diagnosis Guide

*Version 4 Release 1*

**Note!**

Before using this information and the product it supports, be sure to read the general information under “Appendix. Notices” on page 43.

**Fourth Edition, October 2001**

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

Changes to this edition are marked with a vertical bar.

This edition replaces SH12-6337-02.

© Copyright International Business Machines Corporation 1999, 2001. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>About This Book</b> . . . . .	<b>v</b>
Who Should Read This Book . . . . .	v
How This Book Is Organized . . . . .	v

<b>Chapter 1. Log Files and Log Levels of MERVA</b> . . . . .	<b>1</b>
The Windows NT Event Log . . . . .	1
The Diagnosis Log . . . . .	2
The Programmer's Trace Log . . . . .	3
Merging Trace Files . . . . .	5
The Control Process Trace Log . . . . .	5
The Control Process API Trace Log . . . . .	6
Setting the Logging Level . . . . .	7
Trace Information. . . . .	8
Queue Trace Information . . . . .	8
Routing Trace Information. . . . .	8
SWIFT Link Diagnostic Information . . . . .	8
USE Background Process Trace Information . . . . .	9
Maintaining MERVA Log and Trace Files . . . . .	9

<b>Chapter 2. Processing Trace Facility of MERVA Link</b> . . . . .	<b>11</b>
Trace Facility Overview . . . . .	11
Trace Facility Functions . . . . .	12
Trace Level . . . . .	12
Trace File Contents . . . . .	12
PDU Segment Trace Format . . . . .	13
Trace Facilities for Different Process Classes . . . . .	15
Tracing an Inbound SNA APPC Conversation . . . . .	15
Tracing an Inbound TCP/IP Conversation . . . . .	15
Tracing a Sending ASP . . . . .	15
Tracing the MERVA Link Daemon . . . . .	16
Tracing the Remote Card Reader Server . . . . .	16
Trace Facilities for the SNA Server. . . . .	16

<b>Chapter 3. Routing Trace Information</b> . . . . .	<b>17</b>
How to Use a Routing Trace. . . . .	17
Routing Trace Example . . . . .	17
Message Header Information . . . . .	18

Message Body Information . . . . .	19
Reason Codes . . . . .	21

<b>Chapter 4. The Audit Log Database of MERVA</b> . . . . .	<b>25</b>
Database Administration . . . . .	25
Sample Report Programs . . . . .	25
Calling Sequence for Example1 to Example5, and Example7 . . . . .	25
Calling Sequence for Example6. . . . .	26
Calling Sequence for Ctable. . . . .	26
The Message Audit Log . . . . .	27
The User Audit Log . . . . .	28
Table Layouts. . . . .	28
Event Types . . . . .	30
The Customizer Audit Log . . . . .	32
The Client Audit Log . . . . .	33
Event Types . . . . .	34

<b>Chapter 5. Problem Determination</b> . . . . .	<b>35</b>
Initial Evaluation of a MERVA Problem . . . . .	35
Determining Whether the Problem Involves Hardware or Software. . . . .	35
Classifying a Software Problem. . . . .	36
Determining Software Problem Type . . . . .	37

<b>Appendix. Notices</b> . . . . .	<b>43</b>
Trademarks . . . . .	44

<b>Glossary of Terms and Abbreviations</b> . . . . .	<b>47</b>
--	-----------

<b>Bibliography</b> . . . . .	<b>59</b>
MERVA ESA Publications. . . . .	59
MERVA ESA Components Publications . . . . .	59
Other IBM Publications . . . . .	59
S.W.I.F.T. Publications . . . . .	59

<b>Index</b> . . . . .	<b>61</b>
------------------------	-----------



---

## About This Book

Read this book to find out how to diagnose and report problems that can occur with the IBM licensed programs:

- MERVA USE & Branch for Windows NT
- MERVA USE & Branch for Windows NT with SWIFT Link

Both programs are part of MERVA ESA Components Version 4 Release 1. They are referred to as MERVA in this book.

---

## Who Should Read This Book

This book helps system programmers and customer engineers identify and correct problems that occur while they use MERVA. It contains a step-by-step description on how to diagnose and report possible MERVA program failures.

It is assumed that you are familiar with

- Windows NT
- SWIFT network requirements
- Structured Query Language (SQL)
- Procedure Language REXX.

---

## How This Book Is Organized

This book contains the following main chapters:

- Chapter 1 informs you about log files and log levels of MERVA.
- Chapter 2 explains to you the processing trace facility of MERVA Link.
- Chapter 3 tells you how to work with routing trace information.
- Chapter 4 contains information about the audit log database of MERVA.
- Chapter 5 tells you how to determine the cause of a problem.

This book also contains a glossary of terms and abbreviations, a bibliography, and an index.





---

## Chapter 1. Log Files and Log Levels of MERVA

Diagnostic information within MERVA is provided in the form of log files. These log files are located in the directory that is specified with the **PathLog** variable in the file **<instance\_name>.cfg** for the MERVA instance. For more information refer to the *MERVA USE & Branch for Windows NT Installation and Customization Guide*. Diagnostic information helps you locate the cause of problems that might occur.

The following structure shows the directory tree of MERVA log and trace files.

```
----Traces
|
|----Api\enmapi.trc
|----Base\enmbase.trc
|----Qmgt\enmqmgt.trc
|----Routing\enmroute.trc
|----SWIFTLink\enmswift.trc
|----MRVLink\enmmlink.trc
|----Custom\enmcust.trc
|----ExpImp\enmexpim.trc
|----Daemon\enmciapi.trc.<pid>
|       where <pid> denotes the process identifier
|
----Logs
|
|----Base\enmdiag.log
|----Client_DTS\ / ( Name of Partner-LU or TCP/IP hostname).num,
|       where num can be 000 to 999
|----Daemon\enmcidmn.log
```

MERVA logs the trace information in different files according to its components. The following list shows the trace information and an explanation.

<b>Api</b>	MERVA Application Programming Interface
<b>Qmgt</b>	MERVA Queue Management Service
<b>Routing</b>	MERVA Routing Service
<b>SWIFTLink</b>	MERVA SWIFT Communications Service
<b>MRVLink</b>	MERVA Link Service
<b>Custom</b>	MERVA Customization Service
<b>ExpImp</b>	MERVA Import/Export Utility
<b>Daemon</b>	MERVA Control Process
<b>Client-DTS</b>	MERVA Client Data Transfer Service

The directory **Traces\Base** contains general trace entries of MERVA. The directory **Logs\Base** contains the diagnosis log file.

---

### The Windows NT Event Log

The MERVA inetd daemon writes to the Windows NT event log.

MERVA writes entries to the Windows NT event log only during startup when MERVA logging services are not yet started.

To display the Windows NT event log, use the EventLog display of Windows NT.

---

## The Diagnosis Log

The diagnosis log file provides you with information that can:

- Help you recover from error conditions, for example, errors when using the API calls or errors encountered when communicating with other systems (such as a cable not being connected)
- Give you status information about the MERVA Link component
- Give you status information about the SWIFT Link component

The diagnosis log also holds routing information written when the Routing Trace function is selected. See “Chapter 3. Routing Trace Information” on page 17 for more information.

Diagnosis log records are stored in a sequential file called **enmdiag.log**. Each message written to the diagnosis log file consists of two parts, the message header and the message body. The message body starts in a new line.

You can display the diagnosis log file selecting the program group **Diagnosis** and the program **Display Diagnosis Log** from the MERVA Menu program. Alternatively, you can display and search the diagnosis log file using any standard ASCII text editor. Figure 1 shows a sample of records in a diagnosis log file displayed with an ASCII text editor.

```
[ 2010]*157  19990416  13:26:27 ekaasi          checkResetandMIP          MLINK RE
ENM8658E: MIP-Error, received number is not in sequence (21 instead 1)

[ 2012]*199  19990416  13:26:27 ekaasi          MERVA LINK ASI           MLINK RE
ENM8704E: MERVA Link: Error occurred during initialize. Diag:AR2720

local  ASP 'MNTASP'

partner ASP 'MESAASP'

[ 2013]*294  19990416  13:26:27 ekaasi          MERVA LINK ASI           MLINK I
ENM8100E: The explanation of Diag. code AR2720 is:

DC AR2720 is reported by the inbound AS processor EKAASI

RC  39: Error detected during MIP checking

RS  32: Message integrity protocol violation detected

[ 2025]*201  19990416  13:26:38 ekaasi          checkResetandMIP          MLINK I
ENM8705I: MERVA Link [local ASP:'PC63QRK'] received message integrity
          reset indicator from partner [ASP:'QRKPC63']
```

*Figure 1. Example of Diagnosis Log Records*

The layout of the header is as follows:

<b>[Counter]</b>	Represents the record counter for entries in the <b>enmdiag.log</b> file.
<b>Text ID</b>	The first character is an asterisk (*) indicating text.
<b>Length</b>	The length field is five characters long.
<b>Date</b>	The date is in the form of YYYYMMDD, where YYYY is the year, MM is the month, and DD is the day.
<b>Time</b>	The time is in the form HH:MM:SS, where HH is the hour, MM are the minutes, and SS are the seconds.

- Program ID** The program ID is a 15-character code identifying the MERVA module or MERVA library from which the message originates.
- Function ID** The function ID is a 25-character code identifying the function that the message originated from. This field is optional and may not be present.
- Component ID** The component ID is a 5-character code identifying the MERVA component that originates the message.
- Message Console ID** The message console ID is a 2-character code that identifies the message control display.

The layout of the body is as follows:

**Message** The variable-length message to be recorded.

For more information about error messages, refer to *Messages and Codes*. *Messages and Codes* is available online in the documentation folder of MERVA.

**Note:** If you view or list the file by using, for example, a standard ASCII text editor, do not change or save the file. Many editors change the file, for example, they add a special end-of-file character to a file when it is saved.

---

## The Programmer's Trace Log

The programmer's trace log is a general debugging tool. The information it contains should be analyzed by your IBM representative.

Each message written to the programmer's trace log file consists of two parts, the message header and the message body. The message body starts in a new line.

The following figure shows a sample of records in a programmer's trace log file.

```

[ 1]*200 19990818 12:16:04      382 enmcicps      0      0
main          enmcicps.c      192
Named Shared Memory mervaldb/pumlcnt.mem created

[ 2]*192 19990818 12:16:04      382 enmcicps      0      0
main          enmcicps.c      225
Semaphore mervaldb/com_port_sem1 created

[ 3]*190 19990818 12:16:04      382 enmcicps      0      0
main          enmcicps.c      258
Semaphore mervaldb/sem_rc_file created

[ 4]*178 19990818 12:16:04      382 enmcicps      0      0
main()
Program 'enmcicps' started

[ 5]*206 19990818 12:16:06      184 enmcvld      0      0
main()
LogLevel: 1, QueueTrcLevel: 0, RoutingTraceLevel: 0

[ 6]*275 19990818 12:16:07      190 enmcgpum      0      0

ENM5000I: Message counter log status from 1998-08-18 to 1999-08-18 ( current month ).

Current monthly usage ( average ): 0

[ 7]*247 19990818 12:16:07      190 enmcgpum.exe"  0      0
queue_srv
Log-Manager of MERVA Workstation V4.1 succesfully started; queue name = mervaldb/Log_Pipe_File

[ 8]*210 19990818 12:16:09      344 enmcuprc      0      0
main          enmcuilog.c      154
ENP4304I: Background process enmcuprc successfully started

```

Figure 2. Example of Programmer's Trace Log

The layout of the first line of the log entry header is defined as follows:

<b>[Counter]</b>	Represents the record counter for entries in the <b>enmdiag.log</b> file.
<b>Text ID</b>	The first character is an asterisk (*) indicating text.
<b>Length</b>	The length field is five characters long.
<b>Date</b>	The date is in the form YYYYMMDD, where YYYY is the year, MM is the month, and DD is the day.
<b>Time</b>	The time is in the form HH:MM:SS, where HH is the hour, MM are the minutes, and SS are the seconds.
<b>PID</b>	The PID is the process identification given by Windows NT.
<b>MERVA ID</b>	The MERVA ID is an 15-character code identifying either the MERVA component or the MERVA library that the message originated from.
<b>Return code</b>	The return code from the process that created the message.
<b>Reason code</b>	The reason code from the process that created the message.

The layout of the second line of the log entry header is defined as follows:

<b>Function ID</b>	The function ID is a 25-character code identifying the function that the message originated from. This field is optional and may not be present.
--------------------	--

**Called Function ID**

The called function ID is a 25-character code identifying the function that was called within the function identified by function ID. The logged reason code refers to this ID. This field is optional and may not be present.

**File**

The source file name where the error occurred. This field is optional and may not be present.

**Line number**

The source file line number where the error occurred. This field is optional and may not be present.

The layout of the body is defined as follows:

**Message**

The variable-length message that is to be recorded.

**Note:** If you view or list the file by using, for example, a standard ASCII text editor, do not change or save the file. Many editors change the file, for example, they add a special end-of-file character to a file when it is saved.

## Merging Trace Files

As described in “Chapter 1. Log Files and Log Levels of MERVA” on page 1, MERVA creates different trace files. It might, however, be more convenient to combine all trace files in one file. To do this, use the program **enmcgprg.exe**.

**Note:** You have to be a MERVA Administrator to use this program.

To combine all trace files:

1. Set the Windows NT environment variable **ENM\_LOG\_DIR** to point to your logging directory as defined with the **PathLog** variable in the file **<instance\_name>.cfg**. The file **<instance\_name>.cfg** is located in the **instance** directory in your MERVA USE & Branch installation directory.
2. The program **enmcgprg.exe** writes the merged trace file to the current directory. Therefore, change to a directory for which you have write permission.
3. Start **enmcgprg** without any parameters. Enter the complete path for the program. For example, if **enmcgprg** is located in the default installation directory, the complete path is **c:\merva\use\_branch\admin\enmcgprg.exe**.

Then, the trace file **enmplog.log** is created. It contains all trace files.

---

## The Control Process Trace Log

The MERVA Control Process is the component that supervises all MERVA programs and resources.

The records of the control process trace log are stored in a sequential file called **enmcidmn.log**. This file is located in the directory **<PathLog>\Logs\Daemon** where **<PathLog>** denotes the directory that is specified by the **PathLog** variable in the MERVA instance configuration file. The information it contains should be analyzed by your IBM representative.

The control process trace log file can be displayed and searched using any standard ASCII text editor. Figure 3 shows a sample of records in a control process trace log file displayed with an ASCII text editor.

```

19990204 16:00:10:546 ENMDDaemonMain MERVa control process started (PID=182).
19990204 16:00:10:546 ENMDDaemonMain Merva name      : merval
19990204 16:00:10:546 ENMDDaemonMain Instance name   : mervaldb
19990204 16:00:10:546 ENMDDaemonMain Instance owner  :
19990204 16:00:10:546 ENMDDaemonMain IpcDirectory   : merval
19990204 16:00:10:546 ENMDDaemonMain EnvironDirectory : d:\mv4\users\mervaldb
19990204 16:00:10:546 ENMDDaemonMain NLSPATH        : c:\temp
19990204 16:00:10:546 ENMDDaemonMain Logging Directory : c:\temp
19990204 16:00:10:546 RegResource() Name merval/enmcmdqueue found for
alias ENMD_CMD_QUEUE
(requesting process=enmcimai)
19990204 16:00:10:562 RegResource() Name merval/enmdreplyqueue found for
alias ENMD_RPLY_QUEUE
(requesting process=enmcimai)
19990204 16:00:14:640 ENMDDaemonMain StopTimer() has been called
19990204 16:00:14:640 ENMDDaemonMain state: IDLE, command: QUERY.
19990204 16:00:14:640 AnswerQuery() Query(ENMD_QU_LOGLEVEL) replied to PID=150:
iNumber=2 usRc=0.
19990204 16:00:14:640 ENMDDaemonMain StopTimer() has been called
19990204 16:00:14:640 ENMDDaemonMain state: IDLE, command: QUERY.
19990204 16:00:14:640 AnswerQuery() Query(ENMD_QU_LOG_DIRECTORY) replied to PID=150:
string=c:\temp rc=0.
19990204 16:00:14:640 ENMDDaemonMain StopTimer() has been called
19990204 16:00:14:640 ENMDDaemonMain state: IDLE, command: STRT.
19990204 16:00:14:640 ENMDDaemonMain StartTimer(1) has been called
19990204 16:00:14:656 ENMDDaemonMain state: STRUP, command: STRTL.
19990204 16:00:14:656 ENMDDaemonMain StartProcess enmcicps,
program d:\mv4\bin\enmcicps
to be started.
19990204 16:00:14:765 ENMDDaemonMain state: WTSTRT, command: QUERY.
19990204 16:00:14:765 AnswerQuery() Query(ENMD_QU_LOGLEVEL) replied to PID=149:
iNumber=2 usRc=0.
19990204 16:00:14:765 ENMDDaemonMain state: WTSTRT, command: QUERY.
19990204 16:00:14:765 AnswerQuery() Query(ENMD_QU_LOG_DIRECTORY) replied to PID=149:
string=c:\temp rc=0.
19990204 16:00:14:765 ENMDDaemonMain state: WTSTRT, command: QUERY.
19990204 16:00:14:765 AnswerQuery() Query(ENMD_QU_PROCESSNAME) replied to PID=149:
string=enmcimai rc=902.
19990204 16:00:14:765 ENMDDaemonMain state: WTSTRT, command: REGP.
19990204 16:00:14:765 AddToInstanceL Process enmcicps added to instance list.
19990204 16:00:14:765 AddToRunningLi Process enmcicps added to running list.
19990204 16:00:14:796 ENMDDaemonMain state: WTSTRT, command: REGR.

```

*Figure 3. Example Control Process Trace Log in ASCII Text Editor*

The log record is defined as follows:

<b>Date</b>	The date is in the form YYYYMMDD, where YYYY is the year, MM is the month, and DD is the day.
<b>Time</b>	The time is in the form HH:MM:SS:MS, where HH is the hour, MM are the minutes, SS are the seconds, and MS are the milliseconds.
<b>Function ID</b>	The function ID is a 14-character code identifying the function that the message originated from.
<b>Message</b>	The variable-length message that has been recorded.

---

## The Control Process API Trace Log

The MERVa Control Process API is an interface to the MERVa control process for all MERVa programs. It allows programs, for example, to sign on to or sign off from the control process, to request resources, or to query general information.

In a trace log file of a control process API, all data concerning communication between a MERVA process and the control process is recorded. Because this data can be very extensive, the trace log of the control process API is only written if logging level 4 is set.

Each process has its own trace log file. This helps you determine the data that belongs to different processes. The name of the log files is **enmciapi.trc.<pid>** where **<pid>** denotes the process identifier. The process identifier is a unique number that identifies a process. It is assigned by the operating system. The files are located in the directory **<PathLog>\Traces\Daemon** where **<PathLog>** denotes the directory that is specified by the **<PathLog>** variable in the MERVA instance configuration file.

You can display and edit the trace log files of the control process API by using any standard ASCII text editor. The layout of the log entries is identical to the layout of the control process trace log file described in section “The Control Process Trace Log” on page 5.

**Note:** You cannot control the processing of API trace log files with the retention period mechanism. Because many entries are written to the trace log files, their size increases very fast. Ensure that you always have enough disk space left for these files. Ensure also that the log files are deleted from time to time.

If you view or list the file by using, for example, a standard ASCII text editor, do not change or save the file. Many editors change the file, for example, they add a special end-of-file character to a file when it is saved.

---

## Setting the Logging Level

Except for the routing trace, the logging level should only be changed at the request of an IBM representative. Changing the logging level results in more detailed information being gathered for:

- Diagnosis and trace information
- Queue and routing trace information.

You can change the default logging level during customization. In addition, you can change the logging level temporarily while MERVA is running from the **Settings** pull-down of the MERVA menu. You then get the Logging Level window. The following figure shows an example of this window.

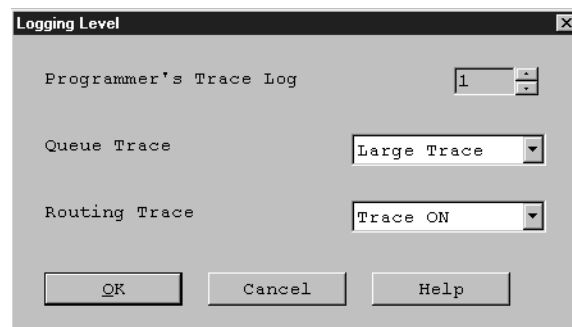


Figure 4. The Logging Level Window of MERVA

**Note:** A large trace, logging level 4, and a logging period of 14 days requires a lot of disk space and might result in filling the hard disk. It also reduces the performance.

## Trace Information

The following logging levels are defined for the control process trace log and the programmer's trace log:

<b>Level 1</b>	Only error information messages are written
<b>Level 2</b>	Level 1 information and additional trace information is written
<b>Level 3</b>	Level 2 information and information messages are written
<b>Level 4</b>	All available information is written.

## Queue Trace Information

The queue trace information specifies how much trace information is written by queue management. The amount of information written to the *enmplog.log* file varies from:

<b>No Trace</b>	No queue trace information is written
<b>Small Trace</b>	A small amount of message trace information is written
<b>Large Trace</b>	The complete message and all related information and fields are written.

## Routing Trace Information

The routing trace information specifies how much routing trace information is written to the *enmdiag.log* file.

The following logging levels are available:

<b>Trace OFF</b>	No routing trace information is written
<b>Trace ON</b>	Routing trace information is activated and written.

---

## SWIFT Link Diagnostic Information

You can create additional diagnostic information for the SWIFT Link component in MERVA. This diagnostic information consists of the following trace files for the Open Systems Interconnection (OSI) layer components:

- ENMTDGPA.DAT
- ENMTAEVC.DAT

These files are located in the directory to which the logging path variable **PathLog** points. The logging path variable is specified at the time the MERVA instance is created.

When you set the logging level for the programmer's trace log to 3 or 4, MERVA writes the trace information to these files.

All trace information is compressed and can only be analyzed by your IBM representative. The trace files are reset at every SWIFT Link startup. To retain a copy of these files, you must rename or print the files before starting the SWIFT Link again.



---

## USE Background Process Trace Information

You can also create diagnosis information for the USE background process component. The diagnosis information is contained in the file **ENNCBAT.DAT**.

This file is located in the directory to which the logging path variable **PathLog** points. The logging path variable is specified at the time the MERVA instance is created.

When you set the logging level for the programmer's trace log to **3** or **4**, MERVA writes the trace information to these files.

All trace information is compressed and can be analyzed only by your IBM representative. The trace file is reset at every MERVA startup. To get a copy of this file, you must rename or copy the file before you restart MERVA.

---

## Maintaining MERVA Log and Trace Files

All MERVA log and trace files that are controlled by the MERVA Log Manager are kept for a specified number of days. This time period is defined during customization. It is called *logging period*. After the specified logging period, the files are renamed to **<file name>.save** where **<file name>** denotes the original log file name. For example, **enmdiag.log** is renamed to **enmdiag.log.save**. New log records are written to a new file. An already existing saved file is deleted.

For more information on the logging period, refer to the *MERVA USE & Branch for Windows NT Installation and Customization Guide*.

The following log files are saved regularly:

- **<PathLog>\Traces\API\enmapi.trc**
- **<PathLog>\Traces\API\Base\enmbase.trc**
- **<PathLog>\Traces\API\Qmgt\enmqmgt.trc**
- **<PathLog>\Traces\API\Routing\enmroute.trc**
- **<PathLog>\Traces\API\SWIFTLink\enmswift.trc**
- **<PathLog>\Traces\API\MRVLink\enmmlink.trc**
- **<PathLog>\Traces\API\Custom\enmcust.trc**
- **<PathLog>\Traces\API\ExpImp\enmexpim.trc**
- **<PathLog>\Logs\Base\enmdiag.log**
- **<PathLog>\Logs\Daemon\enmcidmn.log**

where **<PathLog>** denotes the directory that is specified by the **PathLog** variable in the MERVA instance configuration file.

### Notes:

1. You cannot control the processing of other MERVA log files with the logging period mechanism. Ensure that you have always enough disk space left for these files. Ensure also that the log files are deleted from time to time.
2. Setting the logging period to the maximum period of 14 days requires a large amount of disk space. This might result in the disk being filled. Ensure that you have always enough disk space left for log files.
3. If you view or list the file by using, for example, a standard ASCII text editor, do not change or save the file. Many editors change the file, for example, they add a special end-of-file character to a file when it is saved.



---

## Chapter 2. Processing Trace Facility of MERVA Link

The MERVA Link processing trace facility is supported by MERVA Link programs for diagnosis purposes. This facility writes processing traces to Windows NT files as specified by the customization parameters. A processing trace can be generated by the following processes:

- A process that sends messages via SNA APPC or TCP/IP
- A process that receives message via SNA APPC or TCP/IP
- The MERVA Link daemon
- The Remote Card Reader Server.

---

### Trace Facility Overview

The MERVA Link trace facility parameters determine the Windows NT directory for the trace file of a sending or receiving process and the trace level. The MERVA Link Operating function provides a means to modify these parameters.

A trace file is generated for every instance of a sending Application Support Process (ASP) if requested. The trace file name starts with the name of the ASP.

A trace file is generated for every Windows NT process that handles a MERVA Link inbound SNA APPC conversation if requested. The trace file name starts with the partner LU name for the given conversation. The name of the applicable ASP is not yet known when the trace file is generated.

A trace file is generated for every Windows NT process that handles a MERVA Link inbound TCP/IP conversation if requested. The trace file name starts with the partner TCP/IP host name. The name of the applicable ASP is not yet known when the trace file is generated.

A MERVA Link receiving process is started by the SNA Server or by the TCP/IP services. If it cannot attach the Application Control Table (ACT), the MERVA Link receiving process tries to write a trace to the applicable MERVA IPC directory or to the actual directory. Then the unique trace file names are **ekatpi.trace** and **ekatci.trace**.

The MERVA Link daemon EKAACD supports an activity trace. This trace can be requested by an EKAACD command-line parameter when the daemon is started. The trace file is written to the applicable MERVA file directory. The trace file name starts with **ekaacd**.

The remote card reader server EKACRS supports an activity trace. This trace can be requested by an EKACRS command-line parameter when the remote card reader server is defined as an inetd subserver. The trace file is written to the Windows NT directory that is specified as the second command line parameter. The trace file name is **ekacrs.trace**.

---

## Trace Facility Functions

The trace facility of the MERVA Link processes that handle inbound or outbound messages can be activated and modified using the MERVA Link Operating function. You can specify the Windows NT directory that must hold all trace files and the detail level of the trace to be generated.

If the free space in the file system that contains the trace file directory is not sufficient, no trace is written. A directory entry can be generated. The file size is, however, zero. A failure to write a trace has no influence on handling messages, the main task of a MERVA Link process.

### Trace Level

The amount of information that is written to a trace file is specified by the trace level. A trace level can be specified independently for each of the MERVA Link process groups, the receiving SNA APPC process and the receiving TCP/IP process, and for each sending ASP. The trace level is one of the following numbers:

- 0 No trace must be written. Setting the trace level to zero is the means to switch off tracing for one of the receiving MERVA Link process groups or for a specific sending ASP.
- 1 The process activity must be traced.
- 2 The process activity and the transmitted control information must be traced. The control information includes the Protocol Data Units (PDUs) without the message text.
- 3 The process activity and the transmitted information (PDUs including the message text) must be traced.

The MERVA Link trace facility for inbound and outbound message handling processes is disabled when you reset the trace-directory name to blanks.

### Trace File Contents

The information contained in a MERVA Link processing trace file is mainly the entry and the exit of the MERVA Link programs providing the services of the MERVA Link sublayers (AS, P2, P1, and TP). Other information that can be of interest is written to the trace file, for example:

- Begin of received PDU segments (TPI, TCI)
- Complete PDU segments in hexadecimal and character format (P1I, P1O)
- Error data received from a partner process (TPO, TCO).

Any line in the trace file (trace entry) starts with the name of the MERVA Link program that inserted the trace entry. The program name is followed by a three character identifier of the applicable internal function (for example, EKATPI\_rds and EKAP1I\_cip). The data of a trace entry is self explanatory. Figure 5 on page 13 shows an excerpt from a receiving SNA APPC process with trace level 1.

```

EKATPI      Conversation start on Wed Mar  8 18:57:16 1995
EKATPI      Symbolic partner name is ID0AC291, user ID is HUS
.....

EKATPI_rcv Starting a new PDU   on Wed Mar  8 18:57:16 1995
EKATPI_rds PDU segment received is 0045 0100 , received length is 0045
EKATPI_rds PDU segment received is 0004 81FF , received length is 0004
EKATPI_rds Request for confirmation received
EKATPI_cmp Complete PDU rcvd   on Wed Mar  8 18:57:16 1995

EKAP1I_cip Service Primitive    is ProcessPDU.Indication

EKAP2I_cip Service Primitive    is TEST.Indication, time is 818.412
.....

EKAP2I_tme Activity complete    at 818.578, time delta is 000.166
EKAP2I_trm RC = 00 , RS = 00

EKAP1I_trm RC = 00 , RS = 00

EKATPI_val Transfer confirmation requested by partner TP

EKAP1I_cip Service Primitive    is ProcessPDU.Indication
EKAP1I_trm RC = 00 , RS = 00

EKATPI_cfm Probe or message window confirmed

EKATPI_rcv Starting a new PDU   on Wed Mar  8 18:57:17 1995
EKATPI_rds Deallocated normal received
EKATPI_rcv Conversation deallocated by partner TP

EKAP1I_cip Service Primitive    is DISCONNECT.Indication

EKAP2I_cip Service Primitive    is DISCONNECT.Indication, time is 819.093
.....

EKAP2I_tme Activity complete    at 819.096, time delta is 000.003
EKAP2I_trm RC = 00 , RS = 00

EKAP1I_trm RC = 00 , RS = 00

EKATPI_trm Receiving SNA APPC task is about to terminate

EKATPI      Conversation end   Wed Mar  8 18:57:17 1995
EKATPI      Elapsed time is one second

```

*Figure 5. Receiving SNA APPC Process Trace Example (Excerpt)*

## PDU Segment Trace Format

PDU segments are shown in the processing trace when the trace level is 2 or 3. A PDU segment trace consists of a block of trace entries. The first and the last entry of the block identify the begin and the end of the traced PDU segment (PDU envelope, PDU content, and PDU body).

A PDU segment data-trace entry that displays up to 16 data bytes consists of the following sequence:

1. Trace entry identifier, for example, EKAP1I\_trc
2. Data displacement in hexadecimal, for example, 000010

3. PDU data in hexadecimal character format, four groups of eight hexadecimal characters (0 - 9, A - F)
4. PDU data in ASCII character format enclosed within asterisks.

Figure 6 shows an example of a message envelope and a message heading in a receiving process trace with trace level 2.

```

EKAP10_trc PDU envelope data begin -----*
EKAP10_trc 000000 89000201 1A000110 0C0001A1 E7F7F9F1 * i.....X791 *
EKAP10_trc 000010 F4F8F0C1 0A0002A1 D4D5E3C1 E2D71A00 * 480A...MNTASP.. *
EKAP10_trc 000020 01110B00 01A1D4C5 E2C1D4E3 D50B0002 * .....MESAMTN... *
EKAP10_trc 000030 A1D4C5E2 C1C1E2D7 19000314 0A0001A2 * .MESAASP.....s *
EKAP10_trc 000040 D4D5E3D4 E3D70B00 02A2D4C5 E2C1D4E3 * MNTMTP...sMESAMT *
EKAP10_trc 000050 D7140001 92F3F7F1 F3C1F9F5 F5F2F5F1 * P...k3713A955251 *
EKAP10_trc 000060 C5F0C3F5 F2100001 93F9F9F0 F4F1F3F1 * E0C52...19904131 *
EKAP10_trc 000070 F6F3F0F1 F3050000 B0D50500 01B0F205 * 63013....N....2. *
EKAP10_trc 000080 0002B0D5 050003B0 F2 * ...N....2 *
EKAP10_trc PDU envelope data end -----*

EKAP10_trc PDU content data begin -----*
EKAP10_trc 000000 A3002001 42000210 280001A0 C3969595 * t. ....Conn *
EKAP10_trc 000010 8583A389 969540A3 9640D4C5 D9E5C140 * ection to MERVA *
EKAP10_trc 000020 C5E2C140 4DC84B40 D2968894 8195955D * ESA (H. Kohmann) *
EKAP10_trc 000030 0C0001A1 E7F7F9F1 F4F8F0C1 0A0002A1 * ...X791480A.... *
EKAP10_trc 000040 D4D5E3C1 E2D71A00 02110B00 01A1D4C5 * MNTASP.....ME *
EKAP10_trc 000050 E2C1D4E3 D50B0002 A1D4C5E2 C1C1E2D7 * SAMTN...MESAASP *
EKAP10_trc 000060 14000292 F0F1C5C5 F6F1D2D6 F0F0F0F0 * ...k01EE61K00000 *
EKAP10_trc 000070 F0F1F0F9 0C000392 30314545 36314B4F * 0109...k.....| *
EKAP10_trc 000080 08000492 F0F0F1F8 07000496 F0F0F105 * ...k0018...o001. *
EKAP10_trc 000090 000296E6 050000B0 D5050001 B0F00500 * ..oW....N....0.. *
EKAP10_trc 0000A0 02B0D5 * ..N *
EKAP10_trc PDU content data end -----*

EKAP10_trc PDU body is encoded in EBCDIC format
EKAP10_trc PDU body data begin (complete message text) -----*
EKAP10_trc 000000 08100000 54000000 C0F17AC6 F0F1C9C2 * .....{1:F01IB *
EKAP10_trc 000010 D4C5C4C5 C6C6C1E7 E7E7F0F0 F0F0F0F0 * MEDEFFAXXX000000 *
EKAP10_trc 000020 F0F0F0F0 D0C0F27A C9F1F9F9 C9C2D4C5 * 0000.{2:I199IBME *
EKAP10_trc 000030 C4C5C6C6 C1E7E7E7 E4F1D0C0 F47A0D25 * DEFFAXXXU1.{4:... *
EKAP10_trc 000040 7AF2F07A F10D257A F7F97AD4 85A2A281 * :20:1...:79:Messa *
EKAP10_trc 000050 878540F1 0D2560D0 * ge 1... *
EKAP10_trc PDU body data end -----*

```

Figure 6. Receiving Process PDU Trace Example

MERVA Link PDUs are encoded in EBCDIC. To show readable data of a PDU in the MERVA Link Windows NT processing trace, PDU data is translated to ASCII representation. The MERVA Link Windows NT trace facility uses a specific translation table for that purpose.

The standard EBCDIC characters (numbers 0 - 9, upper and lower case letters, and a number of special characters) are translated to their equivalent in ASCII. The EBCDIC value X'20' (an ASCII blank) is translated to a blank. This means, it is unchanged. All other EBCDIC codes are translated to X'2E', the ASCII representation of a period.

**Note:** In the PDU trace, the numbers are shown in Intel format. The lowest byte is the first byte, the highest byte is the last byte.

---

## Trace Facilities for Different Process Classes

The following sections explain the specific trace facility characteristics of the following process classes:

- Inbound SNA APPC process
- Inbound TCP/IP process
- Outbound process (sending ASP)
- Daemon process
- Remote card reader server.

### Tracing an Inbound SNA APPC Conversation

The MERVA Link program that accepts an inbound SNA APPC conversation is named EKATPI. At the time when it must open the processing trace file it does not know whether it must route the conversation or deliver the received messages. In the latter case, it does also not know the identity of the recipient ASP. This is why the processing trace started by EKATPI is associated with the receiving TP rather than with a logical MERVA Link process (ASP).

The trace file name has the following format:<Partner LU name>.<n>.

<Partner LU name> is the applicable name of the Partner LU of the given conversation.

<n> is a consecutively growing number from '000' to '010'. The program chooses the lowest number available. For example, if numbers '000' and '002' already exist, the next number is '001', not '003'. If <n> exceeds the value '010', it restarts at '000'.

You can set the trace file location in the MERVA Link operating panel. For a detailed description of the trace file location, refer to the *MERVA USE & Branch for Windows NT User's Guide*.

### Tracing an Inbound TCP/IP Conversation

The MERVA Link program that accepts an inbound TCP/IP conversation is called EKATCI. Based on the same rationale as for EKATPI, the processing trace started by EKATCI is associated with the receiving TP.

The trace file name has the following format:<Partner host name>.<n>.

<Partner host name> is the TCP/IP partner hostname of the given connection.

<n> is a consecutively growing number ranging from '000' to '010'. The program chooses the lowest number available. For example, if numbers '000' and '002' already exist, the next number is '001', not '003'.

You can set the trace file location in the MERVA Link operating panel. For a detailed description of the trace file location, refer to the *MERVA USE & Branch for Windows NT User's Guide*.

### Tracing a Sending ASP

The name of a sending ASP trace file is generated dynamically by the MERVA Link program EKAASO (sending application support program). It is the concatenation

of the ASP name, the character string **.t.**, and the time stamp. The format of the time stamp is the same as the time stamp generated by EKATPI.

You can set the trace file location in the MERVA Link operating panel. For a detailed description of the trace file location, refer to the *MERVA USE & Branch for Windows NT User's Guide*.

## Tracing the MERVA Link Daemon

The MERVA Link daemon EKAACD writes an activity trace into a storage area in the ACT header. This area provides space for 16 activity trace entries. Trace entries are written to this area in wrap-around mode.

The trace entries written to the trace area in the ACT header and some additional trace information can be written to a trace file. That trace is requested when the MERVA Link daemon is started. The trace file name is the concatenation of **ekaacd.** and a growing number ranging from '000' to '999'. The format of the time stamp is the same as the time stamp generated by EKATPI.

## Tracing the Remote Card Reader Server

The remote card reader server EKACRS writes an activity trace if a trace is requested by a command-line argument. The second argument can specify the name of the trace directory. When it is specified, a trace is generated.

The name of the trace file is generated dynamically by EKACRS. It consists of the character string **ekacrs.t.xxxxxxxxxx** with:

**xxxxxxxxxx**      10 numeric characters that contain the timestamp when the file is generated

The format of the timestamp is MMDDhhmmss (month of the year, day of the month, hour of the day (0-24), minute of the hour, and second of the minute).

The remote card reader server trace shows the data passed to the card reader and the data received from the card reader in hexadecimal and character representation. The character representation assumes that the character data in the card-reader data-stream is encoded in ASCII.

**Note:** The size of MERVA Link trace files might increase very fast. Ensure that you have always enough disk space for these files. Ensure also that the trace files are deleted from time to time. It is recommended to turn off the MERVA Link trace facility after the problem is solved.

---

## Trace Facilities for the SNA Server

For information about SNA communication traces, refer to the corresponding information for Communications Server or Personal Communication.



---

## Chapter 3. Routing Trace Information

---

### How to Use a Routing Trace

You can use a routing trace during system testing to evaluate whether routing is correct, and whether the routing conditions work as specified. This trace information is written if the **Routing Trace** is set to **Trace ON**. See “Setting the Logging Level” on page 7 for further information.

The routing trace information is written to the diagnosis log file for a specific Message Reference Number (MRN). The routing trace information consists of two parts, the message header and the message body. The header is separated from the message body by an ASCII carriage-return (X'0D') and an ASCII line-feed (X'0A') control character.

The MERV routing function forwards messages from one source queue to one or more target queues based on the routing definitions specified with the MERV Customization program. For a detailed description of the routing concepts, refer to the *MERVA USE & Branch for Windows NT Installation and Customization Guide*.

If the routing definition for a given source queue contains conditional statements, these statements are evaluated sequentially from the first to the last statement, or until a *stop* statement is found. A routing trace entry is written for each evaluated statement that lists the statement used, and the corresponding field contents of the message. You can use this information to determine which routing conditions were evaluated, and why they were true or false.

---

### Routing Trace Example

Figure 7 shows a portion of an ASCII text editor screen with the sample diagnosis log file and routing trace entries displayed.

```

[ 211]*149 19990208 15:29:49 ROUTER      Start
MRN = merva1XX00000001: Route message from queue

[ 212]*170 19990208 15:29:49 ROUTER      Cond. FAILED
MSGOK[OK ] = CANCEL[CANCEL].

----->> No queues added to target list

[ 213]*192 19990208 15:29:49 ROUTER      Cond. FAILED
MSGOK[OK] = OK[OK] AND..

APPL[F] = APC[A].

----->> No queues added to target list

[ 214]*192 19990208 15:29:49 ROUTER      Cond. FAILED
MSGOK[OK] = OK[OK] AND..

APPL[F] = LTC[L].

----->> No queues added to target list

[ 215]*233 19990208 15:29:49 ROUTER      Cond. FAILED
MSGOK[OK] = OK[OK] AND..

APPL[F] = FIN[F] AND..

MSGCAT[1] = MTYP_SYS[0].

----->> No queues added to target list

[ 216]*194 19990208 15:29:49 ROUTER      Cond. CORRECT
MSGOK[OK] = OK[OK] AND..

APPL[F] = FIN[F].

----->> Add to Target List...

[ 217]*142 19990208 15:29:49 ROUTER      Ends OK
MRN = merva1XX00000001: Return target list

```

*Figure 7. Sample Diagnosis Log File with Routing Trace Entries*

**Note:** The square bullet (▪) represents a byte of binary data.

## Message Header Information

The header, for example,

```
[ 211]*149 19990208 15:29:49 ROUTER      Start
```

displays the following information:

<b>[Counter]</b>	Represents the record counter for entries in the <b>enmdiag.log</b> file.
<b>Text ID</b>	The first character is an asterisk (*) indicating text.
<b>Length</b>	The length field is two bytes long.
<b>Date</b>	The date is in the form YYYYMMDD, where YYYY is the year, MM is the month, and DD is the day.
<b>Time</b>	The time is in the form HH:MM:SS, where HH is the hour, MM are the minutes, and SS are the seconds.

- Process ID** An 8-character code identifying the process where the message originated. This field is always filled with the process name ROUTER to indicate the routing trace.
- Router Stage** Represents the current stage during routing trace. The stages are:
- Start
  - Cond. CORRECT
  - WARNING
  - Cond. FAILED
  - Ends OK
  - Ends ERROR.

## Message Body Information

The message body is affected by the current stage of routing trace (Function ID), as shown in Table 1.

Table 1. Message Body Information

Routing Stage	Explanation
<b>Start</b>	<p>The start stage displays the input conditions of the message currently evaluated by the routing component. The following are the conditions displayed using the Start stage:</p> <p><b>No error</b></p> <p>The routing component did not detect an error while analyzing the message. The text displayed resembles the following, where xxxxxxxx is the name of a queue:</p> <pre>[ 200]*149 19990208 15:29:49 ROUTER      Start MRN = merva1XX000000001: Route message from queue</pre> <p><b>Illegal queue name</b></p> <p>The routing component found an illegal queue name in the message. The queue name is not known to the system. This could happen, for example, as a result of corrupted data in the database. The text displayed resembles the following, where xxxxxx is the name of the unknown queue:</p> <pre>[ 201]*149 19990208 15:29:54 ROUTER      Start MRN = merva1XX000000002: Router detects undefined source queue</pre> <p>Messages sent to MBRError</p> <p><b>No default queue specified</b></p> <p>The routing for the current queue has no default target queue defined. This could happen, for example, as a result of corrupted data in the database. Check the routing from this queue using the Customizer. There must be at least one default queue defined. The text displayed resembles the following:</p> <pre>[ 202]*149 19990208 15:29:49 ROUTER      Start MRN = merva1XX000000003: Route detects no default queue</pre> <p>Messages sent to MBRError</p>

Table 1. Message Body Information (continued)

Routing Stage	Explanation
<p><b>Cond. CORRECT</b></p>	<p>This stage displays the checking of the different routing conditions. The following are the conditions displayed using the Cond. CORRECT stage:</p> <p><b>Target added to queue list</b>            The condition evaluated is true and the target queue is added to the target list. The condition is shown as the field, constant, or variable followed by its actual value enclosed in square brackets. If the value is not the expected value, especially for fields, check the definition in the Customizer. The text displayed resembles the following, where xxxxxxxx is the name of the queue added to the target list:</p> <pre>[ 216]*194  19990208  15:29:49  ROUTER          Cond. CORRECT           MSGOK[OK] = OK[OK] AND..           APPL[F] = FIN[F].           -----&gt;&gt; Add to Target List...</pre> <p><b>Target queue is already in list</b>            The condition evaluated is true but the queue is already added to the target list. The text displayed resembles the following, where xxxxxxxx is the name of the queue already in the target list:</p> <pre>[ 216]*194  19990208  15:29:49  ROUTER          Cond. CORRECT           MSGOK[OK] = OK[OK]           -----&gt;&gt; already in Target List...</pre>
<p><b>WARNING</b></p>	<p>The WARNING stage is displayed when a condition is found to be true but the target list already includes four queues. The queue is not added to the list. The text displayed resembles the following:</p> <pre>[ 217]*194  19990208  15:29:50  ROUTER    WARNING           -----&gt;&gt; Target List FULL: no queue added.</pre>
<p><b>Cond. FAILED</b></p>	<p>The routing condition shown is not true. The value of the fields is enclosed by brackets. If the value is not what is expected, especially for fields, check the definition in the Customizer. No action is taken and no queue is added to the target list. The text displayed resembles the following:</p> <pre>[ 215]*233  19990208  15:29:49  ROUTER          Cond. FAILED           MSGOK[OK] = OK[OK] AND..            APPL[F] = FIN[F] AND..            MSGCAT[1] = MTYP_SYS[0].            -----&gt;&gt; No queues added to target list</pre> <p>The router now examines the next condition.</p>

Table 1. Message Body Information (continued)

Routing Stage	Explanation
<b>Ends OK</b>	<p>This stage displays the end of a successful check of routing conditions. The following are the conditions displayed using the Ends OK stage:</p> <p><b>Route to default queue</b>            All the routing condition for this queue have been checked; none have been found to be true, so the message is routed to the default queue defined. The text displayed resembles the following, where xxxxxxxx is the name of the default queue:</p> <pre>[ 217]*142 19990208 15:29:55 ROUTER           Ends OK MRN = merva1XX00000002: No TRUE conds. Route to default</pre> <p><b>Route to target queues</b>            All the routing conditions for this queue have been checked; one or more were found to be true. The target queue list is displayed. If more than four conditions are found to be true, only the first four are displayed. The text displayed resembles the following, where aaaa, bbbb, cccc, and dddd are the names of the target queues:</p> <pre>[ 218]*142 19990208 15:30:00 ROUTER           Ends OK MRN = merva1XX00000007: Return Target List</pre> <p><b>Note:</b> If more than four conditions are found to be true, only the first four are displayed.</p>
<b>Ends ERROR</b>	<p>This stage indicates that an error occurred during routing. The routing component reports the current target queue names found, and the reason code for determining why the routing condition failed. See Table 2 for information on the router return codes.</p> <pre>[ 218]*142 19990208 15:30:05 ROUTER           Ends ERROR Error:  TRN[] = "a"[] MRN = merva1XX000000008: Route to : Router returns ( 416)</pre> <p>The message is sent to the router error queue MBRError.</p>

## Reason Codes

Table 2 contains a list of reason codes and messages, along with a description of each to help you diagnose a problem.

Table 2. List of Reason Codes and Messages

Reason	Description
400	An internal memory problem has occurred. You should restart your system and if the problem persists, contact your IBM representative.
401	The message is currently located in an unknown queue. This may be either a system error as the Customizer should have stopped you deleting a queue containing a message, or the result of database problems. Check your system setup in the Customizer and start the determine whether a database recovery is required using the appropriate DB2 <sup>®</sup> functions.
402	No routing is defined for this queue. This is a system error as the Customizer should automatically create a mandatory default routing for all queues. Check the routing in the Customizer.
403	The first operating in a condition is not a field. Check the routing and field definitions defined in the Customizer.
404	Internal problems occurred when attempting to read the field definition. Restart your system and if the problem persists, contact your IBM representative.

Table 2. List of Reason Codes and Messages (continued)

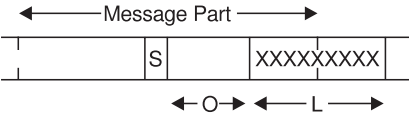
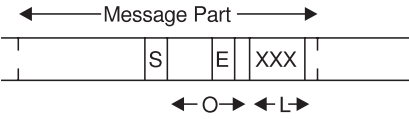
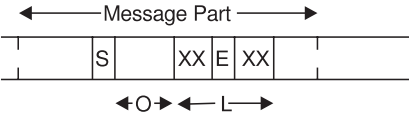
Reason	Description
405	<p>The message exceeds the limits specified for the message part. Check the field definitions specified in the Customizer. If you are using scan patterns, the router has found that the data area specified has exceeded the message part area.</p>  <p>S = Start scan pattern O = Field offset L = Field length X = Field contents</p>
408	<p>The field you defined in the Customizer belongs to an unknown message part. Check the field definition with the Customizer.</p>
409	<p>The field type of the condition has been defined as numeric, but the contents of the field (Operator 1) is nonnumeric. Use the Customizer to change your field definition to "Contains text". See the <i>MERVA USE &amp; Branch for Windows NT Installation and Customization Guide</i> for more detailed information.</p>
410	<p>The field type of the condition has been defined as numeric, but the contents of the constant (Operator 2) is nonnumeric. Use the Customizer to change your field definition to "Contains text". See the <i>MERVA USE &amp; Branch for Windows NT Installation and Customization Guide</i> for more detailed information.</p>
413	<p>The contents of the second operator has been found but does not match the contents allowed. The second operator can be:</p> <ul style="list-style-type: none"> <li>• A predefined constant name</li> <li>• A quoted value</li> <li>• Empty when used with a binary operator, for example, FOUND.</li> </ul>
415	<p>The start of the message exceeds the end scan pattern position. Check the field definitions specified in the Customizer and whether the defined end scan pattern is unique within the message. The router locates only the first occurrence of the pattern.</p>  <p>S = Start scan pattern E = End scan pattern O = Field offset L = Field length X = Field contents</p> <p>This error only occurs when two scan patterns are used.</p>

Table 2. List of Reason Codes and Messages (continued)

Reason	Description
416	<p>The field exceeds the end scan pattern position. Check the field definitions specified in the Customizer and whether the defined scan pattern is unique within the message part.</p>  <p>S = Start scan pattern  E = End scan pattern  O = Field offset  L = Field length  X = Field contents</p>





---

## Chapter 4. The Audit Log Database of MERVA

The logging database holds audit-related log information that can be retrieved using DB2 functions. MERVA uses the following tables to store the information in this database:

- Message Audit Log Table, *enmmsglog*
- User Audit Log Table, *enmualmf*
- User Audit Log Table, *enmualps*
- User Audit Log Table, *enmualof*
- Customizer Audit Log Table, *enmcalog*
- Client Audit Log Table, *enmclualmf*.

---

### Database Administration

Database administration involves making regular backups and deleting old entries to free space in the database. To do this, use the samples listed in “Sample Report Programs” or by backing up and reinstalling the database. See the *MERVA USE & Branch for Windows NT Installation and Customization Guide* for more information on database maintenance.

---

### Sample Report Programs

The following sample programs are available for accessing and obtaining reports on the entries in the logging database:

<b>example1</b>	Report data in the Message Audit Log table <i>enmmsglog</i>
<b>example2</b>	Report data in the User Audit Log table <i>enmualmf</i>
<b>example3</b>	Report data in the User Audit Log table <i>enmualps</i>
<b>example4</b>	Report data in the User Audit Log table <i>enmualof</i>
<b>example5</b>	Report data in the Customizer Audit Log table <i>enmcalog</i>
<b>example6</b>	ISN/OSN reporting example
<b>example7</b>	Report data in the User Audit Log table <i>enmclualmf</i>
<b>cltable</b>	Delete all audit log entries in the log tables.

These programs are written in the programming language REXX and are located in the directory `X:\MERVA_WS\samples\auditlogs` where `X:\MERVA_WS` is the directory where you have installed MERVA.

Note that this directory is write-protected. Therefore, copy the example files to your own directory and run the sample programs from there.

### Calling Sequence for Example1 to Example5, and Example7

The calling sequence for example1 to example5, and example7 is:

```
rexx examplex >filename> (options where x i 1-5, 7
```

The options can be:

**REP=YES** Report with date/time and delimiters

<b>DEL=YES</b>	Copy data to file and delete from table
<b>FIELDS=(all)</b>	All columns are included in the file
<b>FIELDS=(field_list)</b>	Only the columns specified are included in the file; use the column names from the corresponding table
<b>LL=xx</b>	Specifies the line length (can be 80 or 132)
<b>PL=xx</b>	Specifies the page length (default 66)

Example:

```
rex example1 example1.out (fields=WORKSTATION,ISN_OSN DEL=YES
```

## Calling Sequence for Example6

Example6 can be used for ISN/OSN checking. The calling sequence for this program is:

```
example6 <filename> ( options
```

Where the options are:

<b>TYPE=ISN</b>	Report ISN Numbers
<b>TYPE=OSN</b>	Report OSN Numbers
<b>PL=xx</b>	Specifies the page length (default 66)

## Calling Sequence for Ctable

This program is a REXX example to access the MERVA Log database and delete *all* entries from one of the following tables:

- Message Audit Log table enmmmsglog
- Customizer Audit Log table enmcalog
- User Audit Log Message Function table enmualmf
- User Audit Log MERVA Process table enmualps
- User Audit Log all other Function table enmualof
- Client Audit Log table enmclualmf

The calling sequence for this program is:

```
rex ctable /tablename ( options
```

Where the parameters are:

**tablename** is one of the tablenames mentioned above or ALL

**options** The following option is possible:

**/batch** No confirm, just start to delete all entries.

If you want to delete all tables, enter *ALL* as tablename.

Examples:

```
rex ctable /enmmmsglog /batch
```

Deletes all records from the Message Audit Log table.

rex ctable /ALL

Deletes all records from all tables and ask for a confirm.

**Note:** These programs are only samples and can be modified by the user to meet specific requirements. IBM supplies these programs *without* any warranty.

---

## The Message Audit Log

The Message Audit Log records every message immediately after it is sent to or received from the SWIFT network. MERVA Link also records every message to this Log if requested. All log input is stored in the logging database ENMLOGDB. The table in the database that contains the Message Audit Log information is called `enmsglog`.

Table 3 shows the table layout for `enmsglog`.

*Table 3. Table Layout for enmsglog*

Column Name	SQL Type	Length	Description
MSGDATE	DATE	Internal Rep.	The date the message was added to the log database.
MSGTIME	TIME	Internal Rep.	The time the message was added to the log database.
WORKSTATION	VARCHAR	8 bytes	The MERVA instance name in background processes, otherwise the computer name of the used workstation.
PROCESS	VARCHAR	8 bytes	Character code identifying the process that originated the message.
RECEIVING_BIC*	CHAR	12 bytes	The receiving Bank Identifier Code (BIC) of the message.
SENDING_BIC*	CHAR	12 bytes	The sending BIC of the message.
ISN_OSN*	CHAR	6 bytes	The input sequence or output sequence number of the message depending on the type of message.
ACKTYPE	VARCHAR	127 bytes	The acknowledgment identifier for this message.
MRN	CHAR	16 bytes	The message reference number of the message. This is not applicable for SWIFT output messages.
LINKTYPE	CHAR	1 byte	Indicates the message link used to transmit or receive the message. For the current version, this is <b>S</b> for SWIFT Link.
MESSAGE	LONG VARCHAR	32K bytes	The variable-length message recorded (with header information).
CTRL	SMALLINT	2 bytes	Control Information.

**Note:** Internal Rep.: Database internal representation.

\* Relevant for SWIFT messages only.

---

## The User Audit Log

The User Audit Log file contains a record of actions performed while a user was signed on to the system. Those actions can be, for example, signing on to the network or verifying a SWIFT message. The term *user action* includes both, user-driven actions and events triggered by background processes. This includes all the events that change the system status.

### Table Layouts

The user audit log information is divided into the following tables in the database:

- enmualmf** This table includes all User Audit Log information related to message functions within MERVA. The message reference number (MRN), the Bank Identifier Code (BIC), and the message type are recorded. These fields are empty if the information for specific fields is not available.
- enmualps** This table includes all process status information related to processes in the MERVA system, for example, the starting of a program.
- enmualof** All other functions in the system are included in this table. Other functions can be, for example, password changes or transfer of messages from external systems.

The entries in the tables differ according to the information recorded for each message.

Table 4 shows the table layout for enmualmf.

*Table 4. Table Layout for enmualmf*

Column Name	SQL Type	Length	Description
MSGDATE	DATE	Internal Rep.	The date when the message was added to the log database.
MSGTIME	TIME	Internal Rep.	The time when the message was added to the log database.
WORKSTATION	VARCHAR	8 bytes	The MERVA instance name in background processes, otherwise the computer name of the used workstation.
PROCESS	VARCHAR	8 bytes	A character code identifying the process that originated the message.
EVENT_TYPE	CHAR	2 bytes	Event type of this message. See Table 7 for more information about event types.
MRN	CHAR	16 bytes	The Message Reference Number (MRN) of the message added in the table.
BIC*	CHAR	12 bytes	The BIC of the message.
MSG_TYPE	CHAR	4 bytes	Message type number. The APDU message type <i>Ann</i> is recorded in this field.

Table 4. Table Layout for *enmualmf* (continued)

Column Name	SQL Type	Length	Description
USERID	VARCHAR	8 bytes	The identification of the user performing the event recorded.
TEXT	VARCHAR	< 240 bytes	The message that is recorded in the log. This can be variable text.
CTRL	SMALLINT	2 bytes	Control Information.
* Relevant for SWIFT messages only.			

Table 5 shows the table layout for *enmualof*.

Table 5. Table Layout for *enmualof*

Column Name	SQL Type	Length	Description
MSGDATE	DATE	Internal Rep.	The date when the message was added to the log database.
MSGTIME	TIME	Internal Rep.	The time when the message was added to the log database.
WORKSTATION	VARCHAR	8 bytes	The MERVA instance name in background processes, otherwise the computer name of the used workstation.
PROCESS	VARCHAR	8 bytes	A character code identifying the process that originated the message.
EVENT_TYPE	CHAR	2 bytes	Event type of this message. See Table 9 for more information about event types.
USERID	VARCHAR	8 bytes	The identification of the user performing the event recorded.
TEXT	VARCHAR	< 240 bytes	The message that is recorded in the log. This can be variable text.
CTRL	SMALLINT	2 bytes	Control Information.

Table 6 shows the table layout for *enmualps*.

Table 6. Table Layout for *enmualps*

Column Name	SQL Type	Length	Description
MSGDATE	DATE	Internal Rep.	The date when the message was added to the log database.
MSGTIME	TIME	Internal Rep.	The time when the message was added to the log database.
WORKSTATION	VARCHAR	8 bytes	The MERVA instance name in background processes, otherwise the DISPLAY variable of the used workstation.
PROCESS	VARCHAR	8 bytes	A character code identifying the process that originated the message.

Table 6. Table Layout for *enmualps* (continued)

Column Name	SQL Type	Length	Description
EVENT_TYPE	CHAR	2 bytes	Event type of this message. See Table 8 for more information about event types.
USERID	VARCHAR	8 bytes	The identification of the user performing the event recorded.
TEXT	VARCHAR	< 240 bytes	The message that is recorded in the log. This can be variable text.
CTRL	SMALLINT	2 bytes	Control Information.

## Event Types

The following tables show the event types used in the User Audit Log tables.

Table 7. Event Types for Message Functions (*enmualmf*)

Function	Event Type	Description
Create	CC	User created a complete message
	CD	User created and deleted a message
	CI	User created an incomplete message
Edit	EM	User edited and completed a message
	EI	User edited and saved an incomplete message
	ED	User edited and deleted a message
Retype	VM	User retyped and verified a message successfully
	VI	Message rejected
Authorize	AM	User authorized a message
	AI	User did not authorize a message
Complete	UC	Updated or completed an incomplete message
	UD	Updated or deleted an incomplete message
	UI	Updated or left incomplete an already incomplete message
Authenticate	MA	User performed manual authentication
	MI	User did not authenticate message
Delete	DM	User deleted a message
Move	MM	User moved a message
Print	PA	Message printed
MT960/966	MR	Rejected MT960/966 message
	MP	Processed MT960/966 message
API	AA	API added message or API added and routed message
	AD	API deleted message
	AP	API put message or API put and routed message
MERVA Link	RM	Message received from partner
	RR	Status report received and correlated
	SM	Message sent to partner
	SR	Status report sent to partner

Table 8. Event Types for Process Status (enmualps)

Function	Event Type	Description
Automatic Print	AE	End automatic print
	AS	Start automatic print
SWIFT Link	AB	User ended an application, LT or CBT
	LO	User initiated network logout
	NL	User initiated network login
	QU	User quit an application
	SE	User selected an application
Shutdown	SS	User shut down system or user stopped process
Various	RS	Background process started
	RX	Background process stopped
	SP	User started process

Table 9. Event Types for Other Functions (enmualof)

Function	Event Type	Description
Security	AF	User authorized function fault (not authorized)
	PF	User password fault (invalid password entry)
Users	AU	User approved another user
	CP	User changed password
	CF	User updated rights of another user
Correspond.	AC	User deleted entries from correspondent database
	IB	User imported BIC file
	FM	File/database maintenance operation
MERVA Link	ER	Error report message
	ME	Message transfer ended
	RP	Message integrity control file reset
Card Readers	BR	Blacklisted card reader
	DR	User deleted card reader
	RA	User added card reader
	UR	User updated card reader details
ICCs	DS	User deleted ICC set
	FD	User acknowledged receipt of an ICC set
	GB	User added IC card
	GL	User deleted IC card
	IC	User added ICC set
	IU	User maintained user data for IC card
	IW	Update whitelist flag request sent to SWIFT
	NI	Unblocked IC card
	PC	Pin change for IC card
	RC	User read ICC card information
	UF	User updated whitelist flag on IC card
	US	User updated ICC set
	VS	User activated ICC set
SLS	CG	User changed technology flag
	PL	User created pregenerated LOGIN keys
	PN	User created pregenerated SELECT keys
	UK	Update of kernel version for destination

Table 9. Event Types for Other Functions (enmualof) (continued)

Function	Event Type	Description
STK	AK	User activated secure transmission key (STK)
	EK	User entered STK on workstation
	GK	User generated STK key with IC card
	IK	User installed STK on card reader
RSA	GR	User generated RSA key with IC card
Certificates	BV	Revoked certificate for LT
	DV	User deleted certificate from SCR
	MB	Maintained certificate blacklist
	QB	Request of certificate blacklist
	RV	Requested certificate for LT
BKE	BA	BKE was automatically started for correspondent
	BB	User performed backup of bilateral keys
	BC	Increment or reset of BKE counter for destination
	BS	BKE was manually started for correspondent
	CK	User changed authentication keys
	DA	Distributed authentication key information processed
	DD	Authentication key added for destination
	DP	User deleted pre-agreement for correspondent
	DU	Undelete
	PD	Delete pending
	PG	Pre-agreement added for correspondent
	PP	User approved pre-agreement
	PR	Pending reactivate
	PS	Suspend pending
	RB	User restored authentication keys
	SA	Approve suspend/reactivate
	SB	Authentication key for destination discontinued
SD	Authentication key distributed	
UB	User updated authentication key for destination	
UP	User updated pre-agreement for correspondent	
Logon	FN	User failed to logon
	LN	User successfully logged on
Logoff	FF	User failed to logoff
	FU	Administrator forced user
	LF	User successfully logged off

## The Customizer Audit Log

The Customizer Audit Log automatically records any changes saved after using the MERVA customization program.

If the programmer's trace log level is set to 4, all changes made while using the MERVA customization program are also written to the enmplog.log file. The table in the database that contains the Customizer Audit Log is called enmcalog.

Table 10 shows the table layout for enmcalog.

Table 10. Table Layout for enmcalog

Column Name	SQL Type	Length	Description
MSGDATE	DATE	Internal Rep.	The date when the message was added to the log database.



Table 10. Table Layout for enmcalog (continued)

Column Name	SQL Type	Length	Description
MSGTIME	TIME	Internal Rep.	The time when the message was added to the log database.
WORKSTATION	VARCHAR	8 bytes	The WORKSTATION identifier of the workstation that added the message to the table.
PANEL	VARCHAR	15 bytes	A character code identifying the panel from which the log was made.
USERID	VARCHAR	8 bytes	The identification of the user performing the event recorded.
TEXT	VARCHAR	240 bytes	Variable text.
CTRL	SMALLINT	2 bytes	Control Information.

## The Client Audit Log

The Client Audit Log automatically records user actions performed with the Client. The table in the database that contains the Client Audit Log is called enmclualmf.

Table 11 shows the table layout for enmclualmf.

Table 11. Table Layout for enmclualmf

Column Name	SQL Type	Length	Description
MSGDATE	DATE	Internal Rep.	The date when the message was added to the log database.
MSGTIME	TIME	Internal Rep.	The time when the message was added to the log database.
WORKSTATION	VARCHAR	8 bytes	The MERVA AIX (R) instance name in background processes, otherwise the DISPLAY variable of the used AIX station.
PROCESS	VARCHAR	8 bytes	A character code identifying the process that originated the message.
EVENT_TYPE	CHAR	2 bytes	Event type of this message. See Table 12 for more information about event types.
MRN	CHAR	16 bytes	The message reference number of the message.
TARGET_DEST	VARCHAR	34 bytes	Target Destination
MSG_APPL	CHAR	8 bytes	Message Application
MSG_CAT	CHAR	8 bytes	Message Category
MSG_TYPE	CHAR	8 bytes	Message Type of message
MP_FUNCTION	VARCHAR	256 bytes	Message Processing Function
MP_COMPLETION	VARCHAR	256 bytes	Message Processing Completion
SOURCE_QUEUE	CHAR	8 bytes	Source Queue
TARGET_QUEUE	VARCHAR	35 bytes	List of up to 4 Target queues
USERID	VARCHAR	8 bytes	The identification of the user performing the event recorded.

Table 11. Table Layout for *enmclualmf* (continued)

Column Name	SQL Type	Length	Description
CTRL	INT	4 bytes	Control Information.
TEXT	VARCHAR	< 240 bytes	The message that is recorded in the log. This can be variable text.

## Event Types

The following table shows the event types used in the Client Audit Log table.

Table 12. Event Types for Client (*enmclualmf*)

Function	Event Type	Description
Client	YA	Client signon
	YB	Client signoff
	YC	Client put message
	YD	Client got message
	YE	Client replaced message
	YH	Client added message
	YI	Client copied message
	YJ	Client deleted message
	YK	Client moved message
	YL	Client created message
	YM	Client released message
	YN	Client user pressed <b>Authenticate</b> during manual authentication
	YO	Client user pressed <b>Mark as failed</b> during manual authentication
	YQ	Client got template
	YR	Client deleted template
	YS	Client put template
YT	Client released template	
YU	Client replaced template	

---

## Chapter 5. Problem Determination

This chapter tells you how to determine the cause of a problem and describes the documentation required when submitting information to IBM in case of further problem analysis.

If it is necessary to submit an Authorized Program Analysis Report (APAR), you should follow the instructions of your IBM support representative, who will process the APAR through IBM's database of known problems.

---

### Initial Evaluation of a MERVA Problem

An APAR contains a detailed problem description along with any related material you have collected.

If MERVA fails to function correctly, a diagnostic message is displayed. MERVA issues messages starting with the following 3-character prefixes:

<b>ENM</b>	MERVA base and SWIFT Link
<b>ENN</b>	SWIFT USE
<b>ENC</b>	Message Processing
<b>EKA</b>	MERVA Link

An explanation of each message and the action you should take is contained in *Messages and Codes*.

If the same or a related fault occurs after you have taken the recommended action, you must make an initial evaluation of the problem to find out whether it has been caused by an error in IBM-supplied code or by some other error. As problems that occur while MERVA is active can also arise from errors in the operating system or system services, you must also consider these as possible causes.

To make your initial evaluation:

1. Determine if the problem involves Hardware or Software. For further information, see "Determining Whether the Problem Involves Hardware or Software" for a decision.
2. If the problem is software-related, refer to "Classifying a Software Problem" on page 36 for further problem determination.

### Determining Whether the Problem Involves Hardware or Software

Determining whether the problem is related to hardware or software is your first important decision. If you know or suspect that the problem is hardware-related, check the areas mentioned in Figure 8.

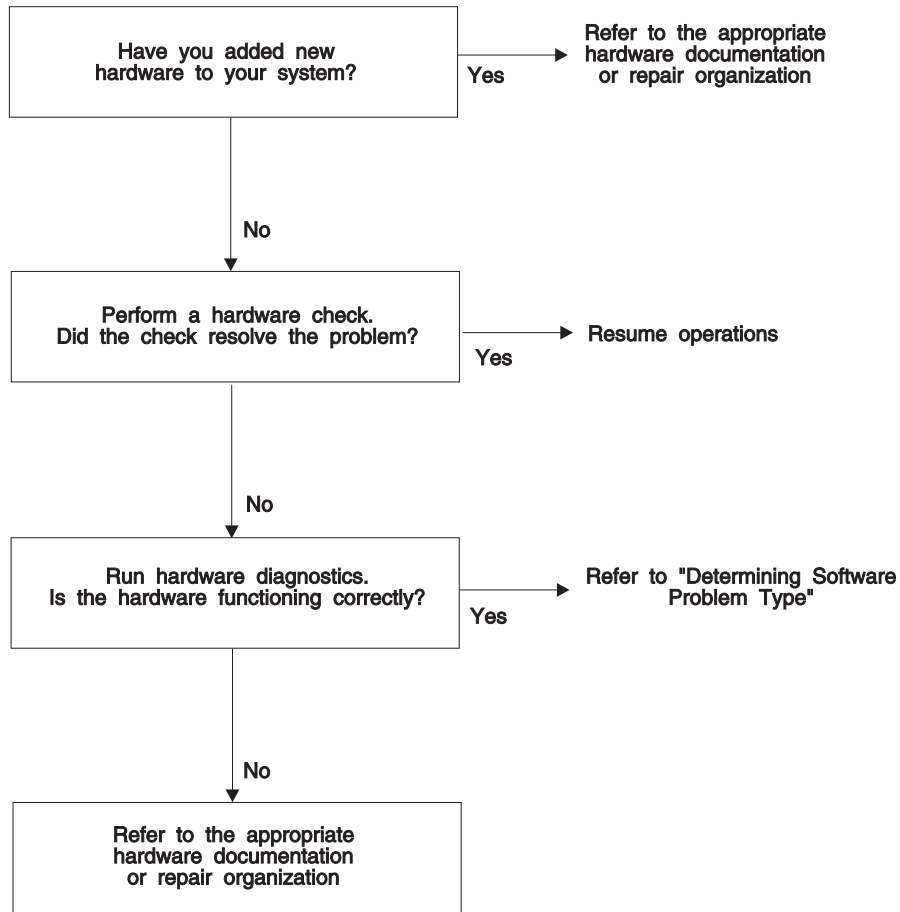


Figure 8. Problem Determination Flowchart

## Classifying a Software Problem

By categorizing a problem by **type** and **component** and reporting this information to IBM, you can narrow the range of probable causes, expedite the search of IBM's Known Problems Database, and quicken the problem resolution process.

- Type refers to the category of symptoms into which the problem falls.
- Component refers to either the operating system, the database and communications software programs, or to a portion of the MERVA software, whichever caused or reported the problem.

## Identifying the Application or Scenario Needed to Re-Create the Problem

The advanced problem-determination procedures presented in this chapter require to re-create the problem to gather diagnostic information to identify the source of the problem.

By answering the following questions, you may isolate the source of the problem. Or, should you later be directed to re-create the problem, this information can be used to help identify the application or scenario needed to reproduce the problem:

1. How would you describe the problem:
  - Specifically, what tasks is the user unable to perform?
  - Is the user's keyboard locked (keys do not respond when pressed)?

2. What was happening when the problem occurred:
  - Was the user working in a Command Prompt window, in a MERVA session, or running a program from the MERVA menu?
  - Did the system stop when the problem occurred?
  - What is the exact sequence of events, including user actions, that led to the problem?
  - Have these actions been carried out successfully in the past? If so, was there anything different about the way or circumstances in which they were attempted when the problem occurred?
  - Reduce the size of the failing application or scenario as much as possible to eliminate extraneous symptoms and to demonstrate the failure more clearly.
  - Were there any indications (messages or otherwise) that a problem might occur?  
If you received any messages, note the message number and any error codes contained in the message. If there is no number associated with a message, record the text of the message.
  - Which was the last operation that you carried out successfully?
  - Which was the last screen that was displayed?
  - Which other programs were active at the workstation?
  - Is it a recurring problem?
3. Are there any special or unusual operating circumstances:
  - Is it a new application?
  - Are new procedures being used?
  - Are there recent changes that might affect the system?
  - Has Windows NT or any other system service been changed since the application or scenario last run successfully?
  - For application programs, which application programming interface (API) was used to create the program?
  - Has a Program Temporary Fix (PTF) recently been installed?
  - Has a CSD, Service Pack, or PTF for another system component been installed recently?

## Determining Software Problem Type

IBM ServiceLink and IBM Support Center databases are structured along the problem categories contained in the following table:

<b>Problem Keyword</b>	<b>Description of the Problem</b>
Installation (INSTALL)	Problems caused by program changes, or experienced during or resulting from the installation of the MERVA program, applications, or new hardware, can be classified as installation problems.
Abnormal termination (ABEND)	Abnormal termination refers to a recurring or unrecoverable problem caused by a program defect that unexpectedly ends processing by a component, application, or function.  An abnormal termination can result from a software compatibility problem or from problems associated with a workstation's hardware configuration.

Problem Keyword	Description of the Problem
Wait or Loop (WAIT/LOOP)	<p>If the user's keyboard is locked (there is no response when pressing any keys) or if the program is not responding as it should, an unexpected program suspension, referred to as a wait, may have occurred.</p> <p>If a program encounters a problem or contains a coding error that causes it to repeatedly execute a certain sequence of instructions while the error condition persists, an uncontrollable program loop may be occurring.</p> <p>Based upon external symptoms, you may not be able to distinguish between a wait and a loop. For this reason, these two types have been grouped together.</p>
Performance problems (PERFM)	<p>Refers to unusually slow or delayed system execution or response time that causes programs to take longer than usual to complete a task. Performance problems can be caused by many factors, including a shortage of disk space, improper allocation of buffers, and Windows NT program resources.</p> <p>A performance problem can also result from a software compatibility problem or from problems associated with a workstation's hardware configuration.</p>
Deficient documentation (DOC)	<p>If the product documentation is deficient in a way that results in lost time for the user (or for other users), report the problem to IBM. Examples of documentation problems include help information that is incorrect or insufficient, or steps or procedures that are inaccurate.</p> <p>If you suspect that the product documentation is in error, describe the documentation error in detail. Include the title and form number of the book (with the page numbers), or the online screen or help containing the erroneous or incomplete information.</p>
Incorrect output (INCORROUT)	<p>Refers to the displaying of unexpected data or the loss or failure of expected data to be displayed.</p> <p>INCORROUT situations are:</p> <ul style="list-style-type: none"> <li>• Output was expected, but not received (missing).</li> <li>• Output was different from what was expected (incorrect).</li> <li>• Data returned by the program is missing or incorrect.</li> </ul>

**Note:** Sometimes external symptoms match more than one problem type. If this occurs, review each description to determine the best match, and then respond as directed for each component.

### **Determining Which Component Is Associated with the Problem**

Use the following descriptions to determine which component reported, caused, or experienced the problem:

#### **Operating system**

Operating system problems include those that occur when responding to command prompts and when using:

- Text editors
- The file system
- The print spooler

If you suspect an operating system problem, refer to “Operating System Problems”.

### **Database**

Database problems include those that occur when using:

- SQL applications
- Database tools

If you suspect a database problem, refer to “Database Problems” on page 40.

### **Communication services**

If you installed MERVA to connect to other systems by means of communications services (that is, using MERVA Link), you might experience communications problems. Communications problems include those that occur when using:

- File transfers
- LAN connections

### **MERVA**

MERVA problems include those that occur when using:

- Functions from the MERVA program menu
- Functions from the MERVA Control Center
- User-written programs that use the MERVA Application Programming Interface (API)

### **Operating System Problems:** Operating system problems include:

- Installation problems (INSTALL)

If you suspect that a user’s installation problem is associated with the operating system, complete the following procedures:

1. Try to re-create the problem.
2. Review the MERVA installation log file **enmwinst.log** located in the **MERVA\USE\_Branch\install** directory.
3. Reinstall Windows NT with the original distribution media. Do not change any files or add any other software. If the operation runs successfully, your problem has been corrected. If not, proceed to step 4.
4. Remove any non-IBM or non-supported IBM hardware and retry the failing operation. If the software problem is resolved, your problem is related to how the hardware interacts with your software. Refer to the information supplied with the hardware and use that information to correct your hardware problem.

- Abnormal termination (ABEND)

Indicators of an abnormal termination include:

- An unexpected return to the Windows NT prompt
- One of the following is displayed on the user’s screen:
  - A message indicating that the system has stopped unexpectedly.  
If you receive such a message, note the occurrence of an abnormal termination.
  - A trap error.

- Wait or loop (WAIT/LOOP)

It may be difficult to tell if an application is blocked, hanging (in an apparently unrecoverable wait state), or looping. If a blockage or hanging has occurred, a wait condition may exist. If you suspect such a condition, attempt to collect information as follows:

1. Try to isolate the problem to a single application. If the problem can be isolated to a single application, suspect that application. Correct the problem using the information supplied with the application.
2. Try to isolate the set of concurrent applications or the circumstances leading up to the wait condition.
3. Document the steps leading to the failure.

If you suspect a loop condition involving the base operating system has occurred, attempt to isolate a specific application that is involved. If the problem can be isolated to a specific application, use the information supplied with the application to correct the problem.

If possible, try to re-create the problem on another system. If the problem cannot be re-created on another system with the same configuration and memory size, suspect a hardware problem.

- Performance Problems (PERFM)
  - Windows NT program components or application programs that can affect other Windows NT components or applications.
  - Hardware restrictions.

In these cases, the following problems may result:

- Unusually slow or delayed system processing or response time that causes programs to take longer than usual to complete a task
- Program suspension when a user switches from one Windows NT session to another Windows NT session
- Unpredictable (incorrect or incomplete) program processing, sometimes resulting in incorrect output
- Program or system failure (abnormal termination).

**Database Problems:** If your problem occurs from within an application program, before proceeding to the problem determination procedures for this type of failure, examine the parameters specified on each Structured Query Language (SQL) command to verify that they have been specified correctly. Refer to *DB2 Building Applications for Windows and OS/2 Environment* for further information.

Database problems include:

- Abnormal termination (ABEND)

**Note:** An abnormal termination can result from a software compatibility problem or from problems associated with your hardware configuration.

Do the following:

1. Write down all the information that was displayed on the user's screen at the time the failure occurred.
2. Note the executable module that reported the abend.
3. If the Independent Trace Facility was active when abnormal termination occurred, use the *db2trc dump* command to write the contents of the Independent Trace Facility trace buffer to a file.



4. If the problem can be re-created, do one of the following actions to gather trace information:

- a. Refer to the preliminary considerations provided in “Using the Independent Trace Facility” in the *DB2 Problem Troubleshooting Guide*.
- b. Simplify the scenario that causes the problem.
- c. Type the following command in the command line:

```
db2trc on -l 0x100000
```

**Note:** Specifying 0x100000 on the -l option of the *db2trc on* command creates a 1 MegaByte trace buffer. If you do not have enough memory, specify a different trace buffer size. It is not the amount of main memory that matters, but rather the amount of main memory plus the potential maximum size of your swap area.

- d. Re-create the problem.
- e. After the problem occurs, enter the following command:

```
db2trc dump filename1
```

where *filename1* is the name of the file that contains the contents of the trace buffer.

- f. Turn off the trace by typing the following command:  

```
db2trc off
```
- g. Contact your IBM Support Center representative.

• Performance problems (PERFM)

Most performance problems can be solved by reorganizing one or more of the MERVA databases:

1. Stop your MERVA system.
2. Stop all DB2 applications that access the MERVA databases you wish to reorganize.
3. Do either of the following:
  - Run the program **enmcwopt.rex**:
    - a. Open a command prompt.
    - b. Change to the **admin** subdirectory of the MERVA installation directory.
    - c. Enter the following command:  

```
rexx enmcwopt <dbname> /V
```

where **<dbname>** is the name of the database to be reorganized (ENMCNTRL, ENMQMSGC, or ENMLOGDB). The option **/V** specifies that the processing information is to contain details.
  - Use the MERVA Control Center:
    - a. Start the MERVA Control Center.
    - b. Double click on Local System.
    - c. Click on the + symbol next to the instance name.
    - d. Select a database.
    - e. Right click on the selected database.
    - f. Select **Optimize dbname**, where *dbname* is the name of the database selected in step 3d.

If the problem persists, perform operating system tuning as described in the Windows NT documentation.

If, after operating system tuning, the problem persists:

1. Record the actual and expected performance along with your reasons for expecting this level of performance (for example, past experience).
2. Contact your IBM Support Center representative.

### **Information Required for Problem Analysis by the MERVA Maintenance Team**

When you report a problem to the MERVA maintenance team, the following information is required:

- Version, release and PTF level of your MERVA system.
- Diagnosis log file **enmdiag.log** and the programmer's trace log.
- File **enmbase.trc** of the error situation if possible with log level 3 or 4.
- MERVA control process log file **enmcidmn.log**.
- Windows NT version number, edition, and service level.
- IBM DB2 UDB 5.2 for Windows NT version number and PTF or CSD level.
- Reference to the MERVA component identifier **5648-B30** and release 1.0.
- SWIFT Link trace file **ENMTDGPA.DAT** in case of SWIFT Link problems. Note that log level 3 is required for this information.
- Corresponding communication error logs and communication configuration files in case of problems. For information about SNA communication traces, refer to the corresponding information for Communications Server or Personal Communication.
- Create a configuration report of MERVA USE & Branch for Windows NT by using the tool **enmnconf**. You can invoke this tool with the command **enmconf outfile**.

---

## Appendix. 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.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

---

## Trademarks

The following terms are trademarks of the IBM Corporation in the United States, other countries, or both:

- Advanced Peer-to-Peer Networking
- AIX
- APPN
- C/370
- CICS
- CICS/ESA
- CICS/MVS
- CICS/VSE
- DB2
- DB2 Universal Database
- Distributed Relational Database Architecture
- DRDA
- IBM
- IMS/ESA
- Language Environment
- MQSeries

- MVS
- MVS/ESA
- MVS/XA
- OS/2
- OS/390
- RACF
- VisualAge
- 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 in the United States, other countries, or both, and is used by IBM Corporation under license.

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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 *SWIFT address*.

**address expansion.** The process by which the full name of a financial institution is obtained using the SWIFT 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 SWIFT 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 SWIFT 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 SWIFT network. Also called a SWIFT 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 SWIFT user institution, or the letters "BIC" for institutions that are not SWIFT 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.** SWIFT 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 SWIFT 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 SWIFT 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 SWIFT field 30, the field tag is :30.

**FIN.** Financial application.

**FIN-Copy.** The MERVA component used for SWIFT 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 SWIFT 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 SWIFT 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 SWIFT 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 2MB 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
- 32KB

**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 SWIFT 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 SWIFT 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 *SWIFT financial message*.

**message body.** The part of the message that contains the message text.

**message category.** A group of messages that are logically related within an application.

**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 header.** The leading part of a message that contains the sender and receiver of the message, the message priority, and the type of message.

**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 reference number (MRN).** A unique 16-digit number assigned to each message for identification purposes. The message reference number consists of an 8-digit domain identifier that is followed by an 8-digit sequence number.

**message sequence number (MSN).** A sequence number for messages transferred by MERVA Link.

**message type (MT).** A number, up to 7 digits long, that identifies a message. SWIFT messages are identified by a 3-digit number; for example SWIFT 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.

**MQA.** MQ Attachment.

**MQ Attachment (MQA).** A MERVA feature that provides message transfer between MERVA and a user-written MQI application.

**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.

**MRN.** Message reference number.

**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, SWIFT MT 195 could be used to request information about a SWIFT MT 100 (customer transfer). The SWIFT MT 100 (or at least its mandatory fields) is then nested in SWIFT 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 SWIFT

**network service access point (NSAP).** The endpoint of a network connection used by the SWIFT 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 SWIFT messages. With NOPROMPT mode, only the SWIFT 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 SWIFT field number to distinguish among different layouts for and meanings of the same field. For example, SWIFT 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 SWIFT network.

**output message.** A message that has been received from the SWIFT 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 SWIFT 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 SWIFT messages. With PROMPT mode, all the fields and tags are displayed for the SWIFT 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 queueing 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 SWIFT messages that are ready for sending to the SWIFT 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 SWIFT 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 signalling 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
- 32KB

**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 SWIFT 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.

**SWIFT address.** Synonym for *bank identifier code*.

**SWIFT 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.

**SWIFT financial message.** A message in one of the SWIFT categories 1 to 9 that you can send or receive via the SWIFT network. See *SWIFT input message* and *SWIFT output message*.

**SWIFT header.** The leading part of a message that contains the sender and receiver of the message, the message priority, and the type of message.

**SWIFT input message.** A SWIFT message with an input header to be sent to the SWIFT network.

**SWIFT link.** The MERVA ESA component used to link to the SWIFT network.

**SWIFT network.** Refers to the SWIFT network of the Society for Worldwide Interbank Financial Telecommunication (S.W.I.F.T.).

**SWIFT output message.** A SWIFT message with an output header coming from the SWIFT network.

**SWIFT system message.** A SWIFT general purpose application (GPA) message or a financial application (FIN) message in SWIFT 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 DSLUNIT and DSLUEND 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 systems 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: you read the message and confirm that you have done so
- Retype verification: 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).

## W

**Windows NT service.** A type of Windows NT application that can run in the background of the Windows NT operating system even when no user is logged on. Typically, such a service has no user interaction and writes its output messages to the Windows NT event log.

## 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 sets.** 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

---

### MERVA ESA Components Publications

- *MERVA Automatic Message Import/Export Facility: User's Guide*, SH12-6389
- *MERVA Connection/NT*, SH12-6339
- *MERVA Connection/400*, SH12-6340
- *MERVA Directory Services*, SH12-6367
- *MERVA Extended Connectivity: Installation and User's Guide*, SH12-6157
- *MERVA Message Processing Client for Windows NT: User's Guide*, SH12-6341
- *MERVA-MQI Attachment User's Guide*, SH12-6714
- *MERVA Traffic Reconciliation*, SH12-6392
- *MERVA USE: Administration Guide*, SH12-6338
- *MERVA USE & Branch for Windows NT: User's Guide*, SH12-6334

- *MERVA USE & Branch for Windows NT: Installation and Customization Guide*, SH12-6335
- *MERVA USE & Branch for Windows NT: Application Programming Guide*, SH12-6336
- *MERVA USE & Branch for Windows NT: Diagnosis Guide*, SH12-6337
- *MERVA USE & Branch for Windows NT: Migration Guide*, SH12-6393
- *MERVA USE & Branch for Windows NT: Installation and Customization Guide*, SH12-6335
- *MERVA Workstation Based Functions*, SH12-6383

---

### Other IBM Publications

- *DB2 Administration Guide*, S10J-8157
- *DB2 Building Applications for Windows and OS/2 Environment*, S10J-8160
- *DB2 API Reference*, S10J-8167
- *DB2 Troubleshooting Guide*, S10J-8169
- *eNetwork Personal Communications Version 4.2 for Windows 95 and Windows NT Quick Beginnings*, GC31-8476
- *eNetwork Personal Communications Version 4.2 for Windows 95 and Windows NT Reference*, GC31-8477
- *CID Enablement Guidelines*, S10H-9666
- *CICS-RACF Security Guide*, SC33-1185
- *ITSC Redbook APPC Security: MVS/ESA, CICS/ESA, and OS/2*, GG24-3960
- *IMS/ESA Version 4 Data Communication Administration Guide*, SC26-3060
- *MQSeries Application Programming Reference*, SC33-1673

---

### 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

## A

administration, database 25  
ASP 15  
audit logs 25

## C

codes, reason 21  
customizer audit log 32

## D

daemon, MERVA Link 16  
database  
    audit log 25  
    enmclualmf 33  
    enmmsglog 27  
    enmualmf 28  
    enmualof 28  
    enmualps 28  
database administration 25  
database problem 40  
delete, log table 26  
diagnosis log  
    contents 2  
    display 2  
    layout 2  
    purpose 2  
    sample 2  
diagnosis trace 7  
diagnostic information 1  
    diagnosis log 2  
    MERVA Link 11  
    SWIFT Link 8  
directory, log 1

## E

enmclualmf 33  
enmmsglog 27  
enmualof 28  
enmualps 28  
evaluation, problem 35  
event log of Windows NT 1  
event types 30  
    enmclualmf 34  
    enmualmf 30  
    enmualof 31  
    enmualps 31

## F

file  
    directory 11  
    ekaacd.trace 11  
    ekacrs.trace 11  
    ekatci.trace 11  
    ekatpi.trace 11  
    enmd.log 5

file (*continued*)

enmdiaq.log 2  
enmdiaq.log.save 3  
enmplog.log 3  
ENMTAEVC.DAT 8  
ENMTDGPA.DAT 8  
ENMX25.DAT 8

## I

inbound conversation 15  
information, status 2

## L

level, logging 7  
log  
    audit logs 25  
    Client audit 33  
    control process 5  
    customizer audit 32  
    diagnosis 2  
    directory 1  
    message audit 27  
    programmer's trace 3  
    user audit log 28  
log file  
    maintaining 9  
log manager 9  
logging level 7

## M

maintaining  
    log file 9  
    trace file 9  
MERVA Link  
    daemon 11, 16  
    diagnostic information 11  
    EKAACD 11  
message  
    body 19  
    header 18  
message audit log 27  
messages 21  
Messages and Codes 3

## N

Notices 43

## O

operating system problem 39

## P

performance problem 41  
problem evaluation 35  
program  
    EKAACD 16  
    EKAASO 15  
    EKACRS 16  
    EKATCI 15  
    EKATPI 15  
programmer's trace  
    contents 3  
    displaying 3  
    layout 4  
    sample 4  
    searching 3

## Q

queue management 8  
queue trace 7

## R

reason codes 21  
remote card reader 16  
report program, sample 25  
routing trace 7  
    contents 17  
    message body 19  
    message header 18  
    sample 18  
    stages 19

## S

sample report programs 25  
sending ASP 15  
SNA  
    APPC conversation 15  
status information 2  
SWIFT Link, diagnostic information 8

## T

table 27, 28  
    enmcalog 32  
    enmclualmf 33  
    enmualmf 28  
    enmualof 29  
    enmualps 29  
TCP/IP conversation 15  
trace  
    control process 5  
    diagnosis 7  
    example 17  
    file 12  
    format 13  
    level 12  
    PDU 13

trace (*continued*)  
  programmer's 3  
  queue 7  
  routing 7, 17  
  Windows NT directory 12  
trace example 18  
trace facility 11  
trace file  
  maintaining 9  
types, event 30

## **U**

user audit log 28

## **W**

Windows NT event log 1

---

## Readers' Comments — We'd Like to Hear from You

MERVA ESA Components  
MERVA USE & Branch for Windows NT  
Diagnosis Guide  
Version 4 Release 1

Publication No. SH12-6337-03

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Complete	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to find	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to understand	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Well organized	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Applicable to your tasks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

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

---

Address

---

Company or Organization

---

Phone No.



Fold and Tape

**Please do not staple**

Fold and Tape

PLACE  
POSTAGE  
STAMP  
HERE

IBM Deutschland Entwicklung GmbH  
Information Development & User Centered Design  
Dept. 0446  
Schoenaicher Strasse 220  
71032 Boeblingen  
Germany

Fold and Tape

**Please do not staple**

Fold and Tape







Program Number: 5648-B30



Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States and other countries.

SH12-6337-03

