MERVA ESA Components

MERVA Automatic Message Import/Export Facility User's Guide

Version 4 Release 1

SH12-6389-02

MERVA ESA Components

MERVA Automatic Message Import/Export Facility User's Guide

Version 4 Release 1

SH12-6389-02

Note

Before using this information and the product it supports, be sure to read the general information under "Appendix H. Notices" on page 77.

Third Edition, May 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.

Contents

About This Book

This manual describes how to install, customize, and use the IBM MERVA Automatic Message Import/Export Facility, hereafter abbreviated to MIE. It is written for users of MIE, for example system support personnel, and assumes you are familiar with all the following:

- Your MERVA system and its connection to networks and other systems
- Your PC and its operating system
- File and message formats

Chapter 1. Introducing MIE

MIE is a program for message transfer through files between your MERVA system and other message-processing applications or systems, such as AccordWorkstation and ECU Netting Workstation. It is a feature of the IBM licensed program MERVA ESA Components.

Overview

MIE consists of:

An automatic message import program
Use this program to start MIE import processes. Each import process searches in a specified directory for message files, and transfers the message files it finds to a specific message queue. Import processes can be started either in batch mode from the command line or, under Windows NT, as service processes.

An automatic message export program
Use this program to start MIE export processes. An export process is triggered by a message export queue, and transfers messages to message files in a specific directory. Export processes can be started either in batch mode from the command line or, under Windows NT, as service processes.

An import and export stop utility
Use this program to stop import or export processes that are running in batch mode.

Import and export process profiles
These profiles let you customize import and export process parameters such as import and export queues, routing conditions, user ID and password, and message directories. Each import and export process has its own process profile (see "Import and Export Process Profiles" on page 27).

A service installation and administration utility
For Windows NT, this program provides an easy way to install, start, terminate, and remove MIE import and export services (see "Appendix F. Service Installation and Administration Utility" on page 55).

A password encryption utility
The password encryption utility lets you set and encrypt passwords stored in process profiles (see "Appendix E. Password Encryption Utility" on page 53).

Concept of Message Transfer

Message transfer can be performed in different ways. Because many products have a file transfer interface, message transfer between MERVA and other message-processing products is done through files.

Files can be exchanged through:

- Local disk if the message-processing products run on the same machine.
- NFS-mounted devices (AIX®).
- Global Local Area Network (LAN) drives if the message-processing products run on different PCs but on a common LAN (Windows NT).

- Diskettes if the message-processing products run on different stand-alone workstations.

With MIE you can automatically:

- Export messages of different types into message files of different formats.
- Import message files of different formats into messages of different types.

MIE also offers you:

- Recovery function

  Messages are not lost and are not transferred twice. If an import or export process ends abnormally during a transfer, the transfer is completed after the next restart of the transfer process.
- Multiple processing

  Several import and export transfers between different directories and queues can be performed at the same time. A separate process profile contains the transfer parameters and the process identification for each process. The respective process owns these process profiles.

The following figure shows you an example for multiple processing of MIE.

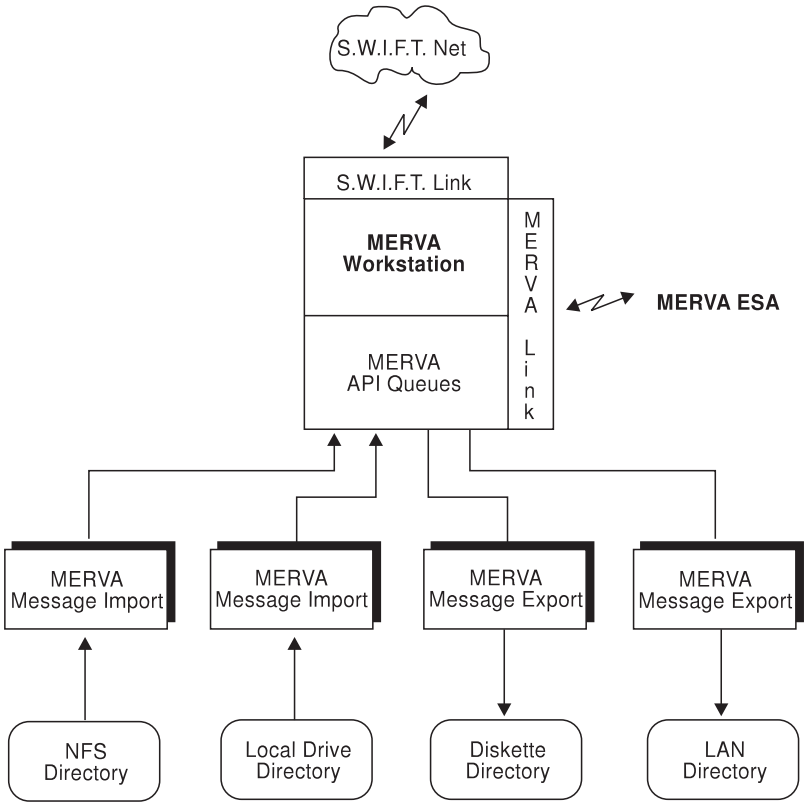

Figure 1. Multiple Message Transfer with MIE

Basic Scenarios

MIE lets you automatically transfer message files between MERVA and other message-processing applications, such as AccordWorkstation and ECU Netting Workstation. The following scenarios show you the possibilities of connecting an AccordWorkstation or an ECU Netting Workstation to MERVA by using MIE.

A MERVA Workstation Scenario

The following figure shows you the SWIFT Link within a MERVA Workstation scenario together with an ACCORD/ECU Netting Workstation. The message application, AccordWorkstation, runs on the same PC or within the same Local Area Network as MERVA Workstation. MERVA Workstation is directly connected to the SWIFT network.

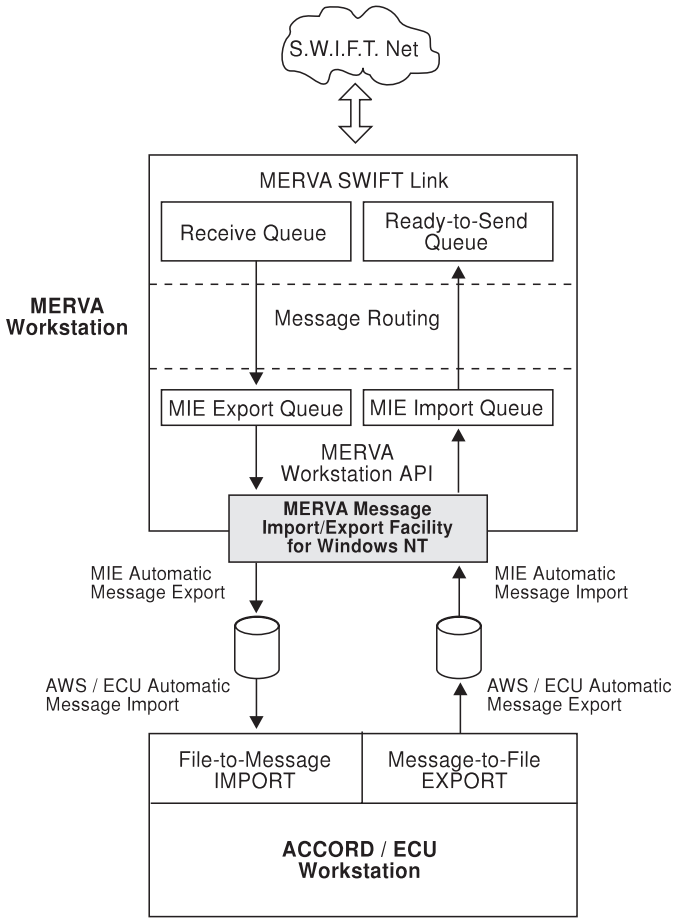

Figure 2. MERVA Workstation Scenario

Examples of routing conditions for this scenario are described in "Routing Conditions for the MERVA Workstation Scenario" on page 51.

A MERVA Connection/NT Scenario

The following figure shows you a message exchange between SWIFT Directory Services Application (DSA) and MERVA. In this scenario, SWIFT DSA and MIE run on the same Windows NT workstation. MERVA, here also called MERVA server, is

located on another Windows NT system. MERVA Connection/NT is used for the communication between MIE and the MERVA server.

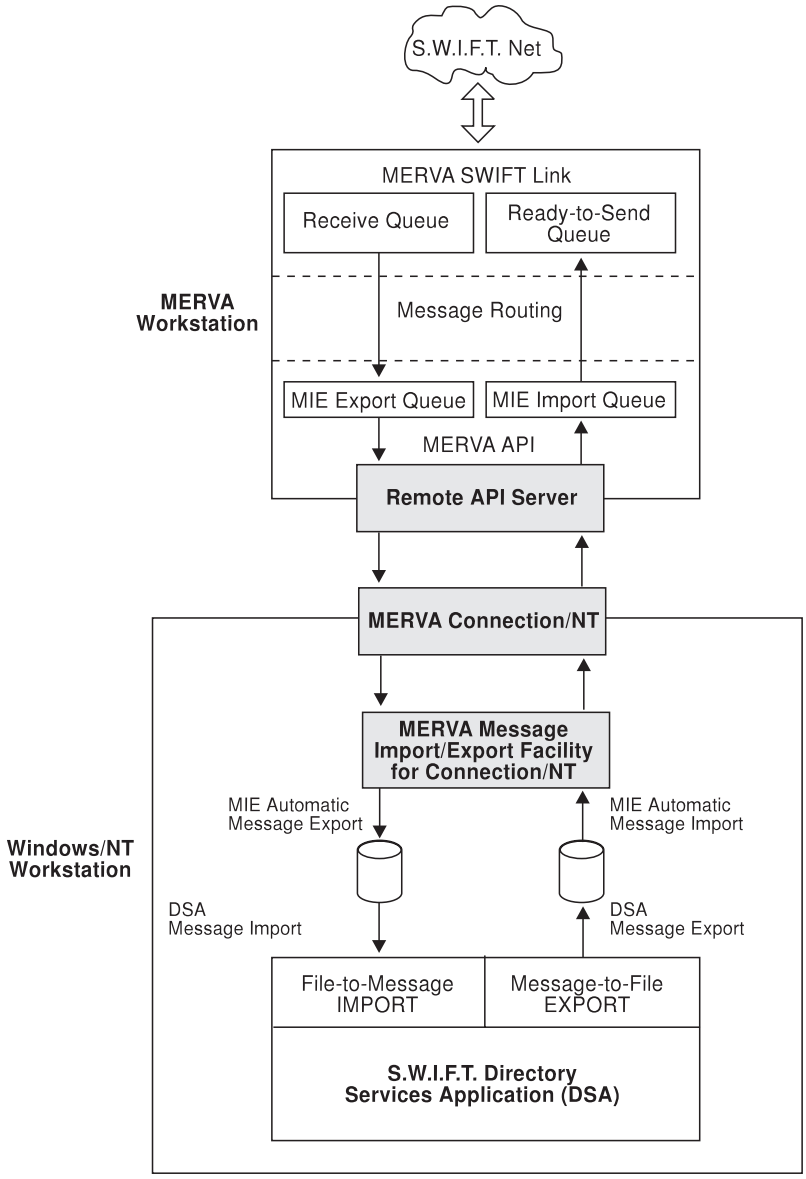

Figure 3. MERVA Connection/NT Scenario

MERVA ESA Front-End Scenario

The following figure shows you a MERVA ESA front-end scenario. The message application, AccordWorkstation, runs on the same PC or within the same Local Area Network as MERVA Workstation. MERVA Workstation is not directly connected to the SWIFT network. It is connected by means of MERVA ESA. A MERVA Link transfers the messages to the SWIFT front-end system. A SWIFT front-end system can also be MERVA Workstation.

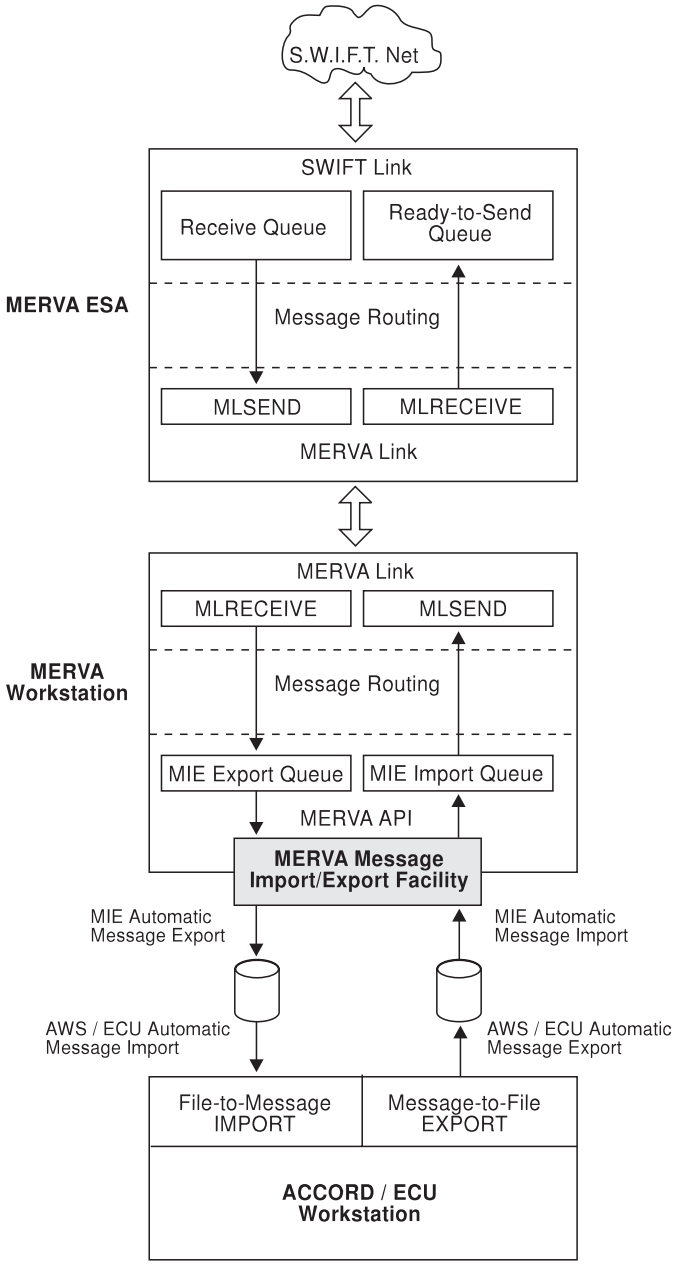

Figure 4. MERVA ESA Front-End Scenario

Examples of routing conditions for this scenario are described in "Routing Conditions for the MERVA ESA Front-End Scenario" on page 51.

MERVA ESA MQSeries NT Scenario

The following figure shows you a MERVA ESA MQSeries NT scenario. It is identical to the MERVA ESA front-end scenario described in Figure 4 except of the communication interface. Instead of MERVA Link and MERVA Connection/NT, MQSeries® is used.

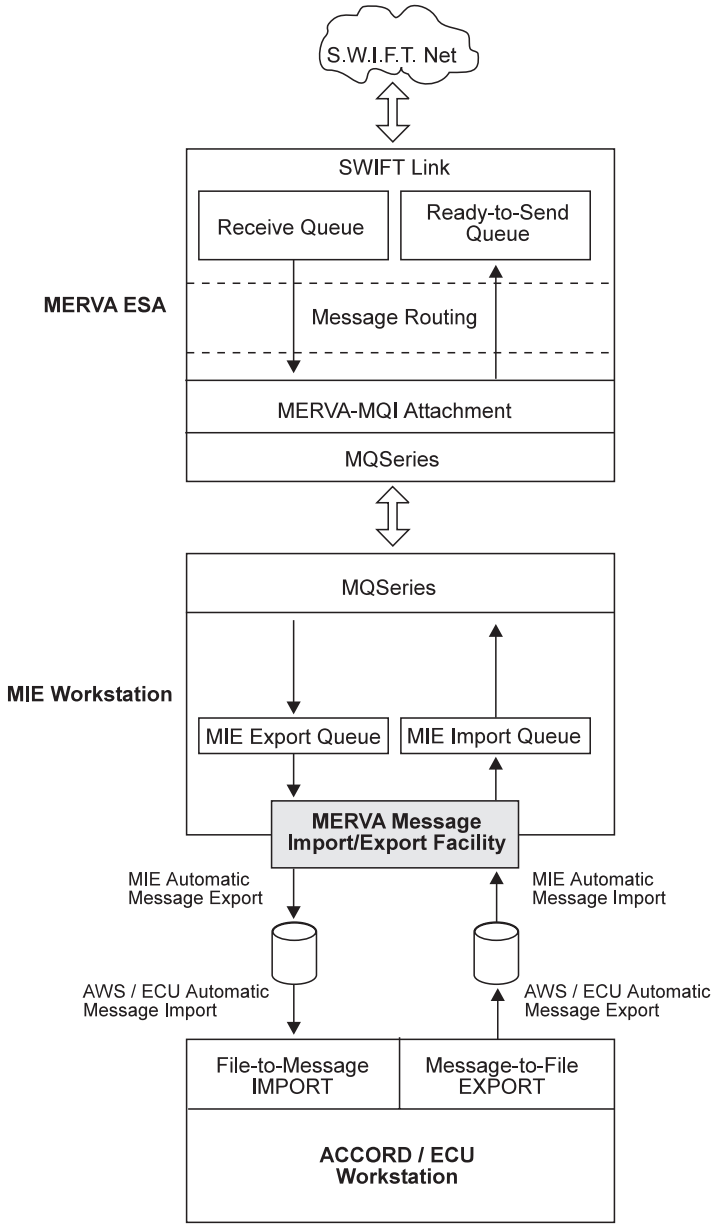

Figure 5. MERVA ESA MQSeries NT Scenario

Message Types and Message File Formats

This section describes the message types and message file formats supported by MIE.

Message Types

The following MERVA message network types are valid:

- SWIFT
- Telex
- Own

MIE supports SWIFT and Own messages. The message type is shown in the message header. You can determine the type for messages that are to be imported. For detailed information on how to do this, refer to “Import and Export Process Profiles” on page 27.

MIE does not support the message type telex explicitly. Telexes are exported in the same way as SWIFT messages. The telex header is not written to the message file. Messages of this kind are reimported as SWIFT or Own messages.

Message File Format

The following formats are valid:

- MERVA format
- AccordWorkstation format

These formats are described in detail in “Appendix C. Structures of Import and Export Message Files” on page 49. How to determine the message file format is described in “Chapter 5. Customizing MIE” on page 23.

Chapter 2. MIE Concepts

This chapter explains in detail how the import and export of message files is organized. The first section describes the import and export with the MERVA API, the second with the MQ Attachment (MQA). It also describes the types and formats of the message files.

Importing Message Files Using MERVA API

This section tells you how the import of message files is organized by using MERVA API.

Architecture of the Import Process

The following figure shows you the basic architecture of the import process.

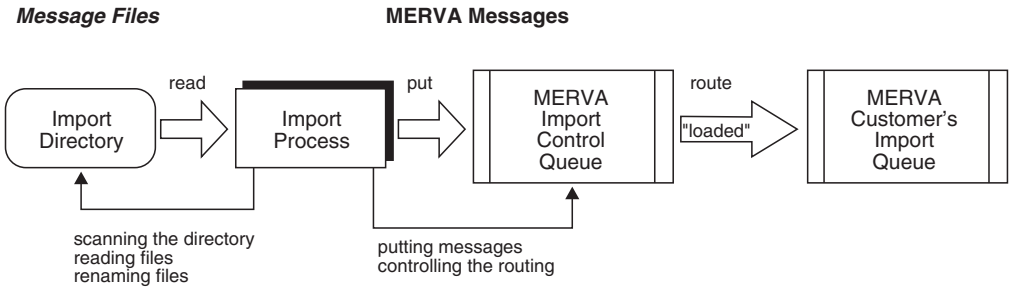

Figure 6. Architecture of the Import Process

Tasks of the Import Process

The import process performs the following tasks:

1. The import directory is searched for message files that are to be imported. The message files are characterized by a file name extension that indicates that these files are to be imported.

   The file names of these message files have the extension **Initial Import File Extension** (refer to the explanation on page 28).
2. The contents of the message files that are found are read, and a format and length check are carried out.
3. If the file is erroneous, the extension of the file name is changed to the **Erroneous File Name Extension** (refer to the explanation on page 29). This file is not processed.
4. If the file is correct, MERVA messages are created, and the contents of the message file are transferred into these messages. These messages are put into the import control queue of the MERVA import process (see **Control Queue Name** on page 31). This state is called *import state 1*.
5. The file name extension of the imported file is then changed to the **Intermediate Import File Extension** (refer to the explanation on page 28). This state is called *import state 2*.
6. The imported messages of the MERVA import control queue are then routed to the customer's import queue with the routing flag *loaded* (see "Constants" on page 25). This state is called *import state 3*.

7. Depending on the value of the **Delete Message File Indicator** (refer to the explanation on page 29), the file name extension of the imported message file is changed to the **Final Import File Extension** (refer to the explanation on page 28), or the file is deleted.

If the import process is interrupted and then restarted with the same process profile, message-processing continues at the last defined import state. This ensures that messages are not lost or processed twice.

You can customize the following import-specific data for each import process:

- Initial Import File Extension
- Erroneous File Name Extension
- Intermediate Import File Extension
- Imported File Extension
- Format of the message file to be imported
- MERVA net type of the imported messages
- Import directory path
- Directory polling frequency
- Name of the MERVA control queue

For information on how you can change this data, refer to "Chapter 5. Customizing MIE" on page 23.

Note: The maximum size of message files to be imported is limited, and depends on the available free dynamic memory space. If the memory is too small, the import process might reject large message files. The maximum message size is limited by the maximum MERVA message size.

Exporting Message Files Using MERVA API

This section tells you how the export of message files is organized.

Architecture of the Export Process

The following figure shows you the basic architecture of the export process using MERVA API.

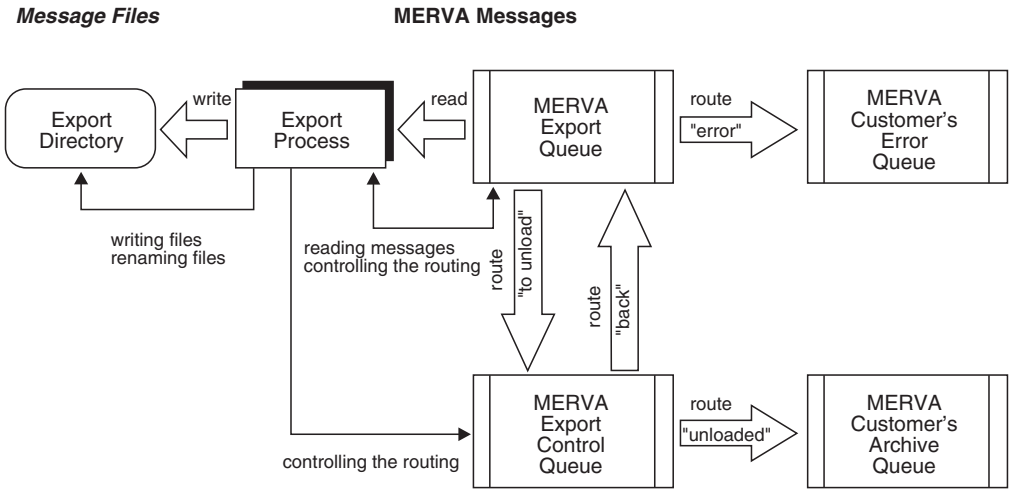

Figure 7. Architecture of the Export Process

Tasks of the Export Process

The MERVA queue manager triggers the export process. When a message enters the MERVA export queue, the export process performs the following tasks:

1. The messages that are found are read.
2. In the export directory, an export message file with the extension **Intermediate Export File Extension** (refer to the explanation on page 29) is created.

   The structure of the file name depends on the value of the parameter **CMEMIE_EX_LONG** (refer to the explanation on page 29). If this parameter is set to the default value **N**, the main part of the file name consists of eight characters of the time stamp *HHmmsshh*. *HH* stands for hours, *mm* stands for minutes, *ss* stands for seconds, and *hh* stands for hundredths of seconds.

   If the value of the parameter **CMEMIE_EX_LONG** is set to **Y**, the prefix *YYYYMMDD* is added to the main part of the file name. *YYYY* stands for the year, *MM* stands for the month, and *DD* stands for the day. The main part of the file name is then *YYYYMMDDHHmmsshh*.

   The messages are transferred to this message file.

   This state is called *export state 1*.
3. The exported MERVA messages get the routing flag *TOUNLOAD* (see "Constants" on page 25) and are routed to the MERVA export control queue. This state is called *export state 2*. If an error occurs during the export process, the corresponding message gets the flag *ERROR* (see "Constants" on page 25). For details see also **Control Queue Name** on page 31.
4. If the maximum number of messages per file is reached (see **Maximum Number of Messages** on page 29), or if the file open time-out applies (see **File Opened Timeout** on page 29), the message file is closed and the file name extension of the export message is changed to **Final Export File Extension** (see the explanation on page 29). This state is called *export state 3*.
5. The exported messages of the MERVA export control queue (see **Control Queue Name** on page 31) get the routing flag *UNLOADED* (see "Constants" on page 25) and are routed to the archive queue.

   In case of a restart, the messages are routed from the export control queue back to the export queue.

If the export process is interrupted and then restarted with the same process profile, message processing continues at the last defined export state. This ensures that messages are not lost or processed twice.

You can customize the following export-specific data for each occurrence of the export process:

- Intermediate Export File Extension
- Final Export File Extension
- Maximum number of messages per message file
- Maximum time-out between opening and closing the message file
- File format of the export message
- Export directory path
- Name of the MERVA export queue
- Alarm semaphore of the MERVA export queue
- Name of the MERVA export control queue

For information on how you can change this data refer to "Chapter 5. Customizing MIE" on page 23.

Importing Message Files Using MERVA-MQI Attachment (MQA)

MERVA-MQI Attachment (MQA) is a subcomponent of MERVA that enables communication between MERVA ESA and MQSeries. For more information, refer to the *MERVA-MQI Attachment User's Guide*.

This section tells you how the import of message files is organized by using MQA.

Architecture of the Import Process

The following figure shows you the basic architecture of the import process.

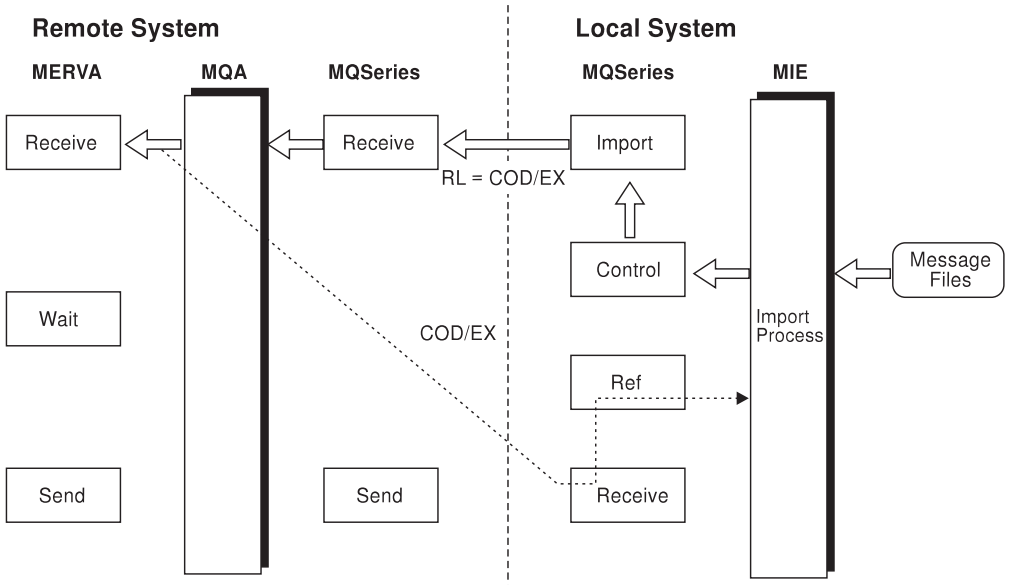

Figure 8. Architecture of the Import Process

Tasks of the Import Process

The import process performs the following tasks:

1. When an import file is found, its contents are read and routed to the control queue as a possible MERVA message. The name of the file is changed to an intermediate file name, such as **xxx.PMI**. The file name specifies that the message is in the control queue.
2. The changes are committed.
3. The imported messages are routed to the import queue. Associated information for each message, such as file name and message ID, is routed to the reference queue for acknowledgment correlation purposes. The file is then renamed to **xxx.MQI**. This file name specifies that MIE is to wait for acknowledgments from the remote system. The changes are then committed. After commitment, MQSeries sends the messages from the import queue to the remote system.
4. MIE handles the next message file.
5. For each received acknowledgment, the associated entry in the reference queue is updated. The changes are immediately committed. This process continues until all messages are acknowledged.
   - If all messages are successfully acknowledged by MQA, the name of the file is changed to the final file name, such as **xxx.IPD**. This file name specifies that the file is imported.
   - If an MQI exception is received for one or more messages, the name of the import file is changed to **xxx.ERR**. The NAK messages are written to a new file with the file extension **xxx.NAK**. This allows you to edit the file and import it later. The message file is imported only partially.

     Note: For the import process with MERVA API, MIE avoids partial import. To avoid this also for the import process with MQI, each file must contain only one message.

Exporting Message Files Using MERVA-MQI Attachment (MQA)

This section tells you how the export of message files is organized.

Architecture of the Export Process

The following figure shows you the basic architecture of the export process using MQA.

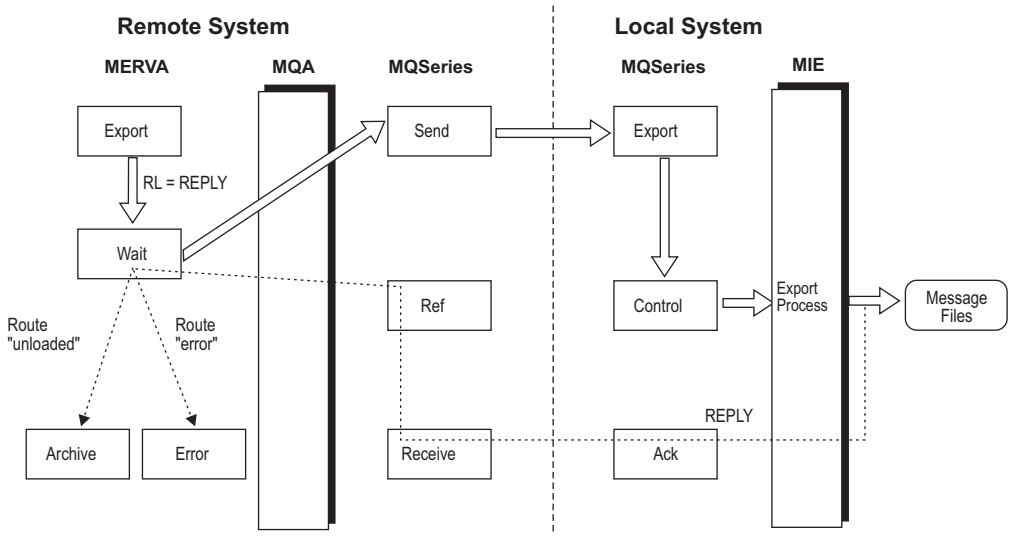

Figure 9. Architecture of the Export Process

Tasks of the Export Process

The export process performs the following tasks:

1. Messages that are to be exported are sent to the export queue on the remote system. They are found by MQA and sent to MIE with the report level **REPLY**.
2. MIE finds the messages in the local export queue, sends them to the control queue and to the intermediate export file. If an error occurs, the message gets the flag **ERROR**.
3. The changes are committed. The exported messages are now in the control queue and in the intermediate export file.
4. The messages are received from the control queue and sent to the ACK queue as reply with the report level **NONE**. If the routing information in the reply is **SUCCESS**, the message is routed to an archive queue in the remote system. If the routing information in the reply is **ERROR**, the message is routed to an error queue in the remote system.
5. At the time specified with **CMECM_EX TOUT** and **CMECM_EX MMIF**, the changes are committed and the name of the export file is changed to the final file name.

After MIE sends the reply for a message to the remote system, no further action is required of MIE. On the remote system, the message waits in the MERVA wait queue or in the MQSeries reference queue until the reply arrives. After the reply arrives, MQ Attachment processes the message accordingly.

Chapter 3. Using MIE

This chapter describes how to start, stop, and use log files to monitor MIE import and export processes. It also describes how you can configure the MIE environment to suit your needs.

Creating Profiles for Import and Export Processes

You need to create one process profile for each import and export process you want to start. Examples of import and export process profiles are contained in the installation directory. If you use MIE with MERVA Connection/NT, you must also provide a MERVA Connection/NT profile, and refer to this profile by setting the variable CMEMIE_IM_CPRF (for an import process) or CMEMIE_EX_CPRF (for an export process) in the process profile.

Starting and Stopping MIE Processes as Batch Jobs

You can run MIE import or export processes as batch jobs when it is not necessary that processes continue to run after you log off Windows NT.

Starting MIE Processes as Batch Jobs

To start an MIE import or export process, do either of the following:

- Double-click on the corresponding start icon in the MIE folder.
- In a Command Prompt window, enter one of the following:
  - **cmecmimp** *profile* (for an import process)
  - **cmecmexp** *profile* (for an export process)

  where ***profile*** is the name of the process profile.

  If necessary, you can also specify either or both of the following parameters:

  -t To turn on the MERVA API trace

  -p To log the customized parameters in the log file

If you use the MERVA interface, and if the MERVA user password is not defined in the process profile, a window similar to the one shown in Figure 10 is displayed.

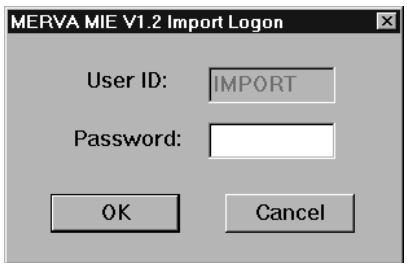

Figure 10. The Import Logon Window

To start the import or export process, enter your password.

If you do not want to have to specify the password each time you start an import or export process, you can do either of the following:

- Specify the password directly in the process profile. The password will not be encrypted.
- Run the encryption utility described in "Appendix E. Password Encryption Utility" on page 53. The password you specify will be stored in the process profile in an encrypted form.

You can start as many import or export processes as you like; however, note that certain parameters must be unique (these parameters are listed in Table 1 on page 18).

Stopping MIE Processes Running as Batch Jobs

MIE import and export processes do not stop automatically. To stop an import or export process, do either of the following:

- Double-click on the corresponding stop icon in the MIE folder.
- Enter Ctrl+C in the Command Prompt window in which MIE is running.
- In any Command Prompt window, enter:
  - **cmecmstp** *profile*

  where ***profile*** is the name of the process profile of the process you want to stop.

Depending on the amount of time the process needs to finish current transfer tasks, and on the values of timers set in the process profile, the process might not be stopped immediately.

Installing, Starting, Stopping, and Removing MIE Processes as Services

You can run MIE import or export processes as services. These services continue to run until you explicitly stop them, even if you log off Windows NT.

Installing MIE Processes as Services

Before you can run an import or export process, you must first install it. To install an MIE process as a service, use the service installation and administration utility, which is provided with MIE and is described in detail in "Appendix F. Service Installation and Administration Utility" on page 55. The parameters that you need to enter with the install command depend on whether you are running MIE so that it communicates with MERVA:

- Directly (QTYP=API)
- Via MERVA Connection/NT (QTYP=CONNECTION)
- Via MQSeries (QTYP=MQI)

The command descriptions in the following sections do not use all of the possible parameters. For a complete list of all parameters, see "Appendix F. Service Installation and Administration Utility" on page 55.

If MIE Communicates with MERVA Directly

If you use the MERVA interface, then a password must be specified in the profile. Enter the following command in a Command Prompt window:

```
cmenmadm.exe -is service_name
     -se service_exe_file
     -sp "profile_file -s service_name"
     -sm message_catalog_dll_file
     -sd MERVA_instance
```

Note that the service name must be specified twice: once immediately after the **-is** keyword, and once immediately after the **-s** keyword.

The following is an example of a typical command to install an MIE import service when MIE communicates with MERVA directly:

```
cmenmadm.exe -is MIE_IMP1 -se cmecmimp.exe -sp "cmecmimp.prf -s MIE_IMP1"
  -sm cmenmmes.dll -sd ENM_merva1
```

If MIE Communicates with MERVA Via MERVA Connection/NT

Enter the following command in a Command Prompt window:

```
cmenmadm.exe -is service_name
     -se service_exe_file
     -sp "profile_file -s service_name"
     -sm message_catalog_dll_file
```

The following is an example of a typical command to install an MIE import service when MIE communicates with MERVA via MERVA Connection/NT:

```
cmenmadm.exe -is MIE_IMP1 -se cmecmimp.exe -sp "cmecmimp.prf -s MIE_IMP1"
  -sm cmenmmes.dll
```

If MIE Communicates with MERVA Via MQSeries

Enter the following command in a Command Prompt window:

```
cmenmadm.exe -is service_name
     -se service_exe_file
     -sp "profile_file -s service_name"
     -sm message_catalog_dll_file
     -sd LUM
```

The following is an example of a typical command to install an MIE import service when MIE communicates with MERVA via MQSeries:

```
cmenmadm.exe -is MIE_IMP1 -se cmecmimp.exe -sp "cmecmimp.prf -s MIE_IMP1"
  -sm cmenmmes.dll -sd "IBM LUM NDL"
```

Starting and Stopping MIE Services

After an MIE service has been installed, you can use the Windows NT Service window to start and stop it just as you would any other Windows NT service. To open this window, select Services from the Control Panel.

Removing (Uninstalling) MIE Services

To remove (uninstall) an MIE process, first stop it, then enter the following command in a Command Prompt window:

```
cmenmadm.exe -rs service_name
```

where *service_name* is the name of the installed service you want to remove.

Configuring the MIE Folder for Windows NT

The initial MERVA folder provides an icon for the import process, an icon for the export process, and two stop icons that contain the respective process profiles as parameters. To add icons for additional import or export processes to your folder:

1. In the folder, copy the icon that you want to change.

2. Select the copied icon.
3. Select **Properties** from the pull-down menu of the selected icon.
4. In the **Target** parameter, change the name of the process profile.
5. Create a new process profile with the name specified in the previous step. If you intend to run more than one process simultaneously, make sure that the following parameters (specified in their profiles) are unique:

Table 1. Parameters that Must Be Unique

For import processes	CMEMIE_IM_APPL	Page 28
	CMEMIE_IM_CPRF	Page 31
	CMEMIE_IM_PATH	Page 28
	CMEMIE_IM_CTRL	Page 31
	CMEMIE_IM_MQRQ	Page 30
	CMEMIE_IM_MQSQ	Page 30
	CMEMIE_IM_MQFQ	Page 30
	CMEMIE_IM_MQCQ	Page 30
For export processes	CMEMIE_EX_APPL	Page 28
	CMEMIE_EX_CPRF	Page 31
	CMEMIE_EX_PATH	Page 28
	CMEMIE_EX_CTRL	Page 31
	CMEMIE_EX_QUEU	Page 31
	CMEMIE_EX_ALRM	Page 31
	CMEMIE_EX_MQRQ	Page 30
	CMEMIE_EX_MQCQ	Page 30
	CMEMIE_EX_MQSQ	Page 30

The following figure shows an example of the folder after its configuration:

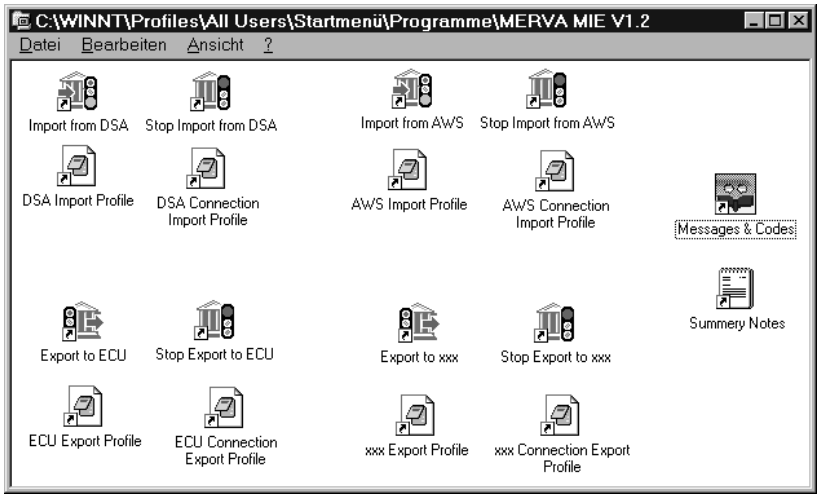

Figure 11. An MIE Folder after Its Configuration

Working with Log Files

MIE processes do not show their activities. Only the MIE log file, which is shared by all import and export processes, gives you information about process activities or errors. The name of the MIE log file is **cmemie.log**, and it is stored in one of the following directories:

- The directory specified by the environment variable **CMEMIELOG**
- If **CMEMIELOG** is not specified, it is stored in the directory specified by the environment variable **HOME**
- If neither **CMEMIELOG** nor **HOME** is specified:
  - If the process is running as a batch job, it is stored in the current directory
  - If the process is running as a service, it is stored in the directory **WindowsNT\System32**

For information about the basic structure of the log file entries and the log messages, refer to *Messages & Codes*. To access *Messages & Codes* for Windows NT, double-click on the appropriate icon in the MIE folder. For a complete list of MIE error codes refer to "Appendix G. Messages and Codes" on page 59.

How to define the extent of the log file information is described in "Import and Export Process Profiles" on page 27.

If an error or an exception occurs, you get a message. The following error situations might occur:

- Critical errors, such as system errors

  An action message indicates that a condition occurred that requires a response from you. The process stops until you close the action message window. In most cases, the process then ends abnormally.
- Non-critical errors, such as message file format error

  An information message indicates that errors occurred and were logged. The process continues. You can then decide when to stop the import or export process and check the log file.

  If you set the log level to the maximum value, the log file contains detailed information. How to set the value for the log level is described in "Import and Export Process Profiles" on page 27.

Tidying Message and Log Files

Completely imported message files of an import directory are renamed according to the **Final Import File Extension** parameter or deleted according to the **Delete Message File Indicator** parameter. How to define these parameters is described in "Import and Export Process Profiles" on page 27.

You can control the log file that is automatically created in the MIE logging directory. You can decide if you want to delete or back up this log file. To do this:

1. Specify the maximum length of the log file in kilobytes.
2. Specify a command that is executed when the specified maximum length is reached. When the log file reaches its maximum length, it is renamed to **cm***MMDDHH***.log**. *MM* stands for the month, *DD* stands for the day, and *HH* stands for the hour. Then, the command that is specified with the parameter **CMEMIE_IM_SAVC** or **CMEMIE_EX_SAVC** is executed.

For details refer to “Import and Export Process Profiles” on page 27, which describes the parameters **CMEMIE_IM_MAXL**, **CMEMIE_EX_MAXL**, **CMEMIE_IM_SAVC**, and **CMEMIE_EX_SAVC**.

Chapter 4. Installing MIE

This chapter tells you how to install the MIE components on your system. Before you install the MIE, install the subsystems of all other products that are used by the MIE. For further information refer to the corresponding installation documentation.

Hardware and Software Requirements

The following tables list the hardware and software requirements needed to install and run MIE:

Table 2. Machine Requirements

Processor	Any Intel x86 compatible processor that can have as its operating system Windows/NT Version 4.0 or a subsequent release
RAM	16MB or more
Hard disk space	20MB or more

Table 3. Basic Software Requirements

Operating system	Windows NT Version 4.0, Service Pack Level 6 or higher
iFOR/LS License Use Management (LUM)	License Use Runtime 4.0.1 (included on the MERVA installation CD-ROM)

Table 4. Additional Software Requirements

If MIE is to communicate with MERVA...	**you also need the following software:**
Directly (QTYP=API)	MERVA USE & Branch for Windows NT Version 4.1
Via MERVA Connection/NT (QTYP=CONNECTION)	MERVA Connection/NT Version 4.1
Via MQSeries (QTYP=MQI)	MQSeries for Windows NT Version 5.1

Installation Steps

On your Windows NT workstation, install MIE in a separate hard disk directory. You can specify the name and drive of this directory. The directory is then created automatically.

To install MIE:

1. Insert the installation CD-ROM in the CD-ROM drive.
2. Start the setup program. To do this, open the Windows NT Explorer and double-click the **SETUP.EXE** in the **\MERVA_Features\Message_Import_Export** directory on the CD-ROM.
3. Follow the instructions that are displayed to you.

The product files are then copied to the directory that you specified during the installation procedure. How to work with these files is described in “Starting and Stopping MIE Processes as Batch Jobs” on page 15.

To open the MIE folder, select **Programs** from the **Start** pull-down menu of Windows NT. The folder looks like the following:

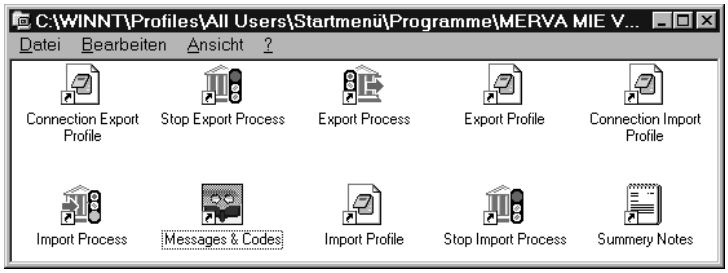

Figure 12. The Initial MIE Folder

Note: The objects in the folder are based on sample data. Before you activate them, check or change the settings of the objects. For information on how to change the settings refer to "Configuring the MIE Folder for Windows NT" on page 17.

Installing the License Keys for MIE

To install the license keys for MIE:

1. Insert the diskette labeled License Key for MERVA MIE.
2. Click **Start → Programs → License Use Runtime → Basic License Tool**.
3. Click **Products → Enroll product**.
4. You then get the Import File window.
5. Click **Import...**
6. Select the file **MIE.lic** from the license key list.
7. Click **OK**.
8. You then get the Details Of window. The license key information is filled in automatically.
9. Click **OK**.

Now, your license key is installed properly.

Chapter 5. Customizing MIE

For the following reasons, customization is necessary:

- The import and export processes are MERVA application programming interface (API) processes and need access to resources of the MERVA API space. The API space should be shared with your applications. For this reason:
  - Define all resources in the MERVA system.
  - Pass the definitions to the corresponding import or export processes.
- The import and export processes need access to system resources, such as directories and semaphores. For this reason:
  - Pass the information about resources and control data to each import or export process.

You must customize the following data:

- MERVA alarms
- MERVA routing parameters
- MERVA routing conditions
- MERVA user IDs
- MIE import and export process profiles

Customizing MERVA Alarms

The MERVA event signalling facility triggers the export process. The signal associated with a queue is called an *alarm*. You must link the alarm semaphore to the MERVA alarm.

Define the name of the alarm semaphore according to the following rules:

- Do not use the substrings **MIE** and **CME**.
- Do not use a MERVA user ID.
- Use unique semaphore names for all export processes.

You must define an alarm name (alarm_name_of_export_queue) for each export process. An example of an alarm name is ALRMEXP.

For more information on how to configure alarms, refer to the appropriate *MERVA Installation and Customization* guide.

Customizing MERVA Routing Parameters

This section describes how to customize the following routing parameters:

- Message queues
- Message fields
- Constants
- Routing conditions

Message Queues

You have to define additional MERVA message queues of the API purpose group according to your naming conventions.

- For each export process you have to define an export queue and an export control queue. You must associate each export queue with an alarm. For details refer to “Customizing MERVA Alarms” on page 23.
- For each import process you have to define an import control queue.

The following table shows you a sample of a customization for two import processes and one export process.

Table 5. Customization of Import and Export Queues for MERVA

Name	Parent Component	Purpose Group	Alarm	Notes
MSGIMPC1	MERVA BASE	API	—	Import Control queue of import process 1
MSGIMPC2	MERVA BASE	API	—	Import control queue of import process 2
MSGEXP	MERVA BASE	API	ALRMEXP	Export queue of export process 1
MSGEXPC	MERVA BASE	API	—	Export control queue of export process 1

For further information on how to associate message queues with alarms, refer to the appropriate *MERVA Installation and Customization* guide.

Message Fields

To route messages between the queues of the import and export processes and your applications, MIE needs a common message field for all import and export messages. By default, the first part of the MERVA **Message Comment** field (**MSGCMNT**) is used to indicate the routing direction. You can also use the **Message OK** field (**MSGOK**) or the **Message Routing** field (**MSGROUTE**). For detailed information refer to the explanation on page 28.

You must define the import and export routing field exactly as shown in the following table, except for the field **Field Name**. In the table that is shown, the name **MIEROUTE** is used. You can, however, choose another name. In this case, you must use the name that is defined by you for all references to this name.

Table 6. Customization of the Message Common Field for MERVA

Field Name	Message Part	Scan Pattern	Offset	Length	Type	Explanation
MIEROUTE	According to the parameter described 28	None	0	8	Text	The field is built to get the routing constants as described in “Constants” on page 25 as its values.

For further information on how to define message fields, refer to the appropriate *MERVA Installation and Customization* guide.

Constants

MIE needs common constants for all import and export messages that are used as routing flags and that represent the values of the routing fields.

You must define the evaluation for the constants exactly as shown in the following table, except for the field **Name**. For this field, you can choose other names. In this case, you must use the names that are defined by you for all references to these names.

Table 7. Customization of the Routing Constants for MERVA

Name	Evaluation	Identification Flag
MIE_ERRO	"ERROR "	Erroneous messages
MIE_LOAD	"LOADED "	Successfully imported messages
MIE_TOUL	"TOUNLOAD"	Messages to be exported
MIE_UNLD	"UNLOADED"	Successfully exported messages
MIE_BACK	"UNLDBACK"	Recovered export messages

Note: You must fill unused portions of the evaluation of the constant with blanks up to the string length of eight.

For further information on constants, refer to the appropriate *MERVA Installation and Customization* guide.

Routing Conditions

MIE requires additional routing conditions. You can determine where processed import and export messages are to be routed, for example, exported messages can be routed to:

- A queue where you can delete the messages manually by using the MERVA function **Expunge**.
- The MERVA target queue **MBDELETE** where the messages are deleted automatically.

For further information on how to delete messages, refer to the appropriate *MERVA Installation and Customization* guide.

Note that MIE does not route messages that are to be exported to the export queue of your system. You must route these messages by means of your applications.

The routing conditions of the import and export processes must match the routing configurations as described in Figure 6 on page 9 and Figure 7 on page 11. The following table shows you how to customize the routing conditions with respect to the process configurations that are shown in Table 5 on page 24, Table 6 on page 24, and Table 7. Note that the queue names are examples. You can change them as you like.

For AccordWorkstation or ECU Netting Workstation, the import queue can be the SWIFT Link ready-to-send queue or the MERVA Link send queue.

Table 8. Customization of the Routing Conditions for MERVA

Messages in queue ...	Are sent by default to ...	Except when ...	Then the messages are routed to ...	Further routing
MSGIMPC1	MSGIMPC1	MIEROUTE = MIE_LOAD	Your import queue of import process 1	None
MSGIMPC2	MSGIMPC2	MIEROUTE = MIE_LOAD	Your import queue of import process 2	None
MSGEXP	MSGEXP	MIEROUTE = MIE_ERRO	Your error queue of export process	None
		MIEROUTE = MIE_TOUL	MSGEXPC	
MSGEXPC	MSGEXPC	MIEROUTE = MIE_UNLD	Your archive queue of export process	None
		MIEROUTE = MIE_BACK	MSGEXP	

For further information on how to set up routing conditions, refer to the appropriate *MERVA User's Guide*.

For examples of routing conditions for the scenarios described in "A MERVA Workstation Scenario" on page 3 and in "A MERVA Connection/NT Scenario" on page 3, refer to "Appendix D. Examples of MERVA Routing Conditions" on page 51.

MERVA User IDs for the Import and Export Processes

MIE needs a MERVA user ID to connect to MERVA. Use the appropriate MERVA program to create the user IDs for all import and export processes. With the user IDs you can set up access rights to:

- **Display and Print — Messages by Queue** to all MIE queues that are triggered by the import and export processes
- Your application program

Note: The MERVA user ID should not contain the substrings **MIE** and **CME**.

For further information on how to set up routing conditions, refer to the appropriate *MERVA User's Guide*.

Customizing MERVA ESA Routing Conditions

You have to adapt the routing conditions of the MQA wait queue to MIE export process. To do this, you have to define the routing conditions relating to the field **MSGACK**. This field is set by MIE export process to "UNLOADED" or to "ERROR ".

You should define the routing conditions in the following way:

- If the field **MSGACK** is set to "UNLOADED", the message is routed to an archive or delete queue.

- If the field **MSGACK** is set to "ERROR ", the message is routed to an error queue.

For more details on how to set routing conditions for MERVA ESA and how to configure MQI, refer to the *MERVA ESA Customization Guide*.

Customizing Process Profile Values and Parameters of MIE

This section describes how to customize the MIE process profiles, and the rules that apply to them.

Import and Export Process Profiles

Each import and export process has a process profile that delivers MERVA customization data and control information. A process profile consists of a flat file with values. Each value is located on a separate line. Characteristics of these values are:

- Each value starts with a tag that is followed by an equal sign (=) and a tag value.
- The last character of each line must be one of the following:
  - A carriage return followed by a line feed
  - A line feed followed by a carriage return
  - A carriage return
  - A line feed

The number sign (#) designates the comment character. Each character succeeding the number sign is considered as comment.

For a detailed description of the syntax of import and export process profiles, refer to "Appendix A. Import and Export Process Profile Structures" on page 33.

The following list describes the process profile values in detail:

General Parameters

These are the general values for the process profile:

Log Level (CMEMIE_IM_LOGL or CMEMIE_EX_LOGL)
This parameter specifies the log level for the import and export process. Valid values are:

1	Only errors are logged.
2	Errors and warnings are logged.
3	Errors, warnings, and information messages are logged.
4	Errors, warnings, information messages, and additional report information, such as each message transfer, are logged.

If you set the log level to the maximum value, the log file contains the most detailed information. This might help you with troubleshooting.

Note: Each process starts with log level **4** until the log level in the process profile is read.

Maximum Length of Log File (CMEMIE_IM_MAXL or CMEMIE_EX_MAXL)
This parameter specifies the maximum length of the log file in kilobytes.

When the length of the log file reaches the specified value, the log file is renamed. At the same time, the command that is specified with the parameters **CMEMIE_IM_SAVC** or **CMEMIE_EX_SAVC** is executed. If the maximum length of the log file is set to zero, the length of the log file is not checked. The value of this parameter can be from zero up to 1000000.

Log File Save Command (CMEMIE_IM_SAVC or CMEMIE_EX_SAVC)
With this parameter you specify the command that is executed when the length of the log file reaches the value that is specified with **CMEMIE_IM_MAXL** or **CMEMIE_EX_MAXL**.

You can pass the following parameters up to 20 times:

%1 This is a placeholder for the fully qualified name of the log file, for example, **c:\merva\mie\cm102913.log**.

%2 This is a placeholder for the fully qualified name of the log file, but without an extension, for example, **c:\merva\mie\cm102913**.

For example, for Windows NT you might specify:

```
CMEMIE_IM_SAVC = cmd /c ren %1 %2.sic
```

Note that you cannot execute command line commands directly. Use **cmd /c** to have the command interpreter execute the command.

Application Name (CMEMIE_IM_APPL or CMEMIE_EX_APPL)
This parameter defines the name of an import or export process. Each name must be unique. The default is the value of the corresponding MERVA user ID (CMEMIE_IM_IDEN or CMEMIE_EX_IDEN; see page 30). The value of this parameter must be unique among all running processes.

Import and Export Directory (CMEMIE_IM_PATH or CMEMIE_EX_PATH)
This parameter indicates the fully qualified path to the directory in which message files to be imported are located, or in which exported message files are to be created. The value of this parameter must be unique among all running processes.

Polling Timeout (CMEMIE_IM_POLL)
This parameter specifies the time-out between two consecutive polling activities of the import process in seconds. The time range is from zero to 3600. A polling activity takes place when an import process searches in the import directory for message files to be imported.

Routing Message Field (CMEMIE_IM_MFLD or CMEMIE_EX_MFLD)
This parameter specifies the message field that MIE uses to route messages. If this parameter is not set, the **MSGCMNT** field is used. Allowed values are **MSGCMNT**, **MSGOK**, and **MSGROUTE**.

Note: Do not use **MSGOK** if you use the MQI interface.

Initial Import File Extension (CMEMIE_IM_FEXT)
This parameter specifies that only files that have this file name extension are to be imported from the import directory. After the import process accesses the files, the initial import file extension is changed into the intermediate import file extension.

Intermediate Import File Extension (CMEMIE_IM_IEXT)
This parameter determines the file name extension of files that are currently accessed by the import process. After the import process is finished, the intermediate import file extension is changed into the final import file extension.

Final Import File Extension (CMEMIE_IM_PEXT)
This parameter determines the file name extension of files that are successfully imported.

Intermediate Export File Extension (CMEMIE_EX_IEXT)
This parameter determines the file name extension of files that are currently accessed by the export process. After the export process is finished, the intermediate export file extension is changed into the final export file extension.

Final Export File Extension (CMEMIE_EX_FEXT)
This parameter determines the file name extension of message files that are successfully exported.

Extension of Erroneous Import Files (CMEMIE_IM_EEXT)
This parameter determines the file name extension of message files that cannot be imported. After an error occurs, the file extension is changed into the extension of erroneous import files.

Long File Names (CMEMIE_EX_LONG)
This parameter specifies the name format of the exported files.

If this parameter is set to the default value **N**, the main part of the file name consists of eight characters of the time stamp *HHmmsshh*. *HH* stands for hours, *mm* stands for minutes, *ss* stands for seconds, and *hh* stands for hundredths of seconds.

If the parameter is set to **Y**, the prefix *YYYYMMDD* is added to the main part of the file name. *YYYY* stands for the year, *MM* stands for the month, and *DD* stands for the day. The main part of the file name is then *YYYYMMDDHHmmsshh*.

Message File Format (CMEMIE_IM_MFIL or CMEMIE_EX_MFIL)
This parameter determines the format of the import and export message files. If the parameter is set to **M**, the type of the message file format is MERVA. If the parameter is set to **A**, the type of the message file format is AccordWorkstation. For detailed information refer to "Appendix C. Structures of Import and Export Message Files" on page 49.

MERVA Message Header (CMEMIE_IM_NTYP)
This parameter determines the MERVA header type for imported messages. If the parameter is set to **S**, the type of the message header is SWIFT If the parameter is set to **O**, the type of the message header is Own. For details about message headers refer to the appropriate *MERVA Installation and Customization* guide.

Delete Message File Indicator (CMEMIE_IM_DELF)
This parameter determines whether the message file is deleted after it is imported. If the parameter is set to **Y**, the file is deleted. If the parameter is set to **N**, the file is not deleted, and the file name extension is changed into the final import file name extension.

File Opened Timeout (CMEMIE_EX_TOUT)
This parameter determines the time interval between opening and closing an export file in seconds. After this time, the current message file is closed regardless of whether the maximum number of messages is reached. The value for the time interval can be from zero to 3600.

Maximum Number of Messages (CMEMIE_EX_MMIF)
This parameter determines the maximum number of exported messages in

one export message file. When the specified number is reached, the file is closed. The value can be from 1 to 1000.

Connection Type (CMEMIE_IM_QTYP or CMEMIE_EX_QTYP)
This parameter specifies the type of the queuing system. The value can be one of the following:

API MIE communicates directly with the local MERVA. This is the default value.

CONNECTION
MIE communicates with MERVA via MERVA Connection/NT.

MQI MIE communicates with MERVA via MQSeries and MERVA-MQI Attachment. If you use this connection type, ensure that the maximum number of uncommitted messages for the queue manager is sufficient to handle the maximum number of messages that an import or export message file can contain.

MERVA User ID (CMEMIE_IM_IDEN or CMEMIE_EX_IDEN)
If the value of the Connection Type parameter (CMEMIE_IM_QTYP or CMEMIE_EX_QTYP) is:

- **API** or **CONNECTION**, an import or export process must log on to MERVA and access MERVA resources, so the value of this parameter must be a MERVA user ID. This user ID, as well as its password and access rights, must be defined in MERVA. The default values are IMPORT (for CMEMIE_IM_IDEN) and EXPORT (for CMEMIE_EX_IDEN).
- **MQI**, the value of this parameter is written to the User Identifier field of the MQSeries message. If the value of CMEMIE_IM_IDEN or CMEMIE_EX_IDEN is null, the user ID of the process that started MIE is assumed as the default.

Parameters for Connection Type MQI

These are the parameters for the process profile when using the connection type MQI; that is, when MIE communicates with MERVA via MQSeries and MERVA-MQI Attachment.

MQSeries Queue Manager Name (CMEMIE_IM_MQQM or CMEMIE_EX_MQQM)
This parameter specifies the name of the local queue manager. If this parameter is not specified, the default queue manager is used.

MQSeries Receive Queue (CMEMIE_IM_MQRQ or CMEMIE_EX_MQRQ)
This parameter defines the name of the MQA receive queue. The value of this parameter must be unique among all running processes.

MQSeries Send Queue (CMEMIE_IM_MQSQ or CMEMIE_EX_MQSQ)
This parameter defines the name of the MQA send queue. The value of this parameter must be unique among all running processes.

MQSeries Reference Queue (CMEMIE_IM_MQFQ)
This parameter defines the name of the MQA reference queue. The value of this parameter must be unique among all running processes.

MQSeries Status Queue (CMEMIE_IM_MQCQ or CMEMIE_EX_MQCQ)
This parameter defines the name of the MQA status queue. The value of this parameter must be unique among all running processes.

File Extension when MIE waits for MQA acknowledgment (CMEMIE_IM_MEXT)
This parameter defines the type of the import file for which MIE waits for an acknowledgment.

Extension of files containing NAK messages (CMEMIE_IM_NEXT)
This parameter defines the type of the import file to which MIE writes the messages that cannot be imported. These messages are called NAK messages.

Remote System Specification (CMEMIE_IM_MQVS or CMEMIE_EX_MQVS)
This parameter defines whether the remote system is a Virtual Storage Extended (VSE) system (Y) or any other system. If the remote system is a VSE system, the remote MQSeries system does not have to send an exception report.

Parameters for Connection Types API and CONNECTION

These are the parameters for the process profile when using the connection types API or CONNECTION; that is, when.MIE communicates with MERVA either directly or via MERVA Connection/NT.

MERVA User Password (CMEMIE_IM_PASS or CMEMIE_EX_PASS)
If you do not want to be prompted for your password each time you start a process, you can store your password in the process profile of the process. The MIE logon window is not displayed, and the process starts without prompting you for your password.

If encrypting your password is not necessary, specify your password directly; otherwise, run the encryption utility described in "Appendix E. Password Encryption Utility" on page 53. After your password has been encrypted, the parameter CMEMIE_IM_PENC (for an import profile) or CMEMIE_EX_PENC (for an export profile) is added to the profile to indicate that the password has been encrypted.

Control Queue Name (CMEMIE_IM_CTRL or CMEMIE_EX_CTRL)
This parameter passes the name of the MERVA control queue to the export process. The value of this parameter must be unique among all running processes.

Export Queue Name (CMEMIE_EX_QUEU)
This parameter passes the name of the MERVA export queue to the export process. The value of this parameter must be unique among all running processes.

Export Queue Alarm (CMEMIE_EX_ALRM)
This parameter passes the alarm semaphore name that is associated with the MERVA export queue to the export process. The value of this parameter must be unique among all running processes.

Connection Profile (CMEMIE_IM_CPRF or CMEMIE_EX_CPRF)
This parameter applies only to the connection type CONNECTION. It specifies the full path of the profile name that is used to connect to the MERVA server machine. The value of this parameter must be unique among all running processes.

Customization Data Check

When the import and export processes read the customization data from the import and export process profiles, the following rules apply:

- All file extensions of one process profile must be different.

- All queue names of one process profile must be different.
- The MERVA user ID must not conflict with internal semaphore names.
- The semaphore name of the export queue alarm must not conflict with internal semaphore names.
- The semaphore name of the export queue alarm must not conflict with the MERVA user ID.

Note: MIE does not perform cross-checking for all import and export files.

Therefore, you should ensure that the following rules apply:

- Assign a unique MERVA application name to each import and export process.
- Assign a unique MERVA user ID to each import and export process.
- Do not use MERVA queue names or the corresponding alarms of an import or export process for any other process.
- Do not use conflicting message file extensions within the same directory.
- Ensure that the import and export directories are clearly separated.
- Ensure that the MERVA customization exists, because its parameters are passed to the import and export processes, and if it does not exist, the missing data (queues, routing data, and other data) will lead to runtime errors in the import and export processes.

Appendix A. Import and Export Process Profile Structures

An import or export profile is a flat file in which you can define one parameter per line. The line format is:

tag **=** *value* **#** *comments*

Leading or intermediate spaces and tabs are ignored. If the tag is not recognized, the line is ignored. If a tag occurs in more than one line, the value of the last occurrence is taken. The tags in the process profile can be in arbitrary order.

Tags and Values for the Import Process Profile Structure

The following table lists the:

- Tags of the import process profile
- Data types of the tag values
- Default tag values that are taken if the corresponding tags are missing or not valid

Note that the type description is divided into numbers and characters. The number specifies the maximum length of the tag values. The character **c** specifies that the value must be a character string. The character **n** specifies that the value must be an integer number.

Table 9. The Import Process Profile Structure

Tag	Type	Explanation	Default
CMEMIE_IM_LOGL	1n	Page 27	1
CMEMIE_IM_MAXL	7n	Page 27	100
CMEMIE_IM_SAVC	256c	Page 28	
CMEMIE_IM_APPL	8c	Page 28	value of CMEMIE_IM_IDEN
CMEMIE_IM_IDEN	8c	Page 30	IMPORT
CMEMIE_IM_PASS	8c	Page 31	
CMEMIE_IM_CPRF	80c	Page 31	imp_con.prf
CMEMIE_IM_PATH	80c	Page 28	
CMEMIE_IM_POLL	4n	Page 28	60
CMEMIE_IM_CTRL	8c	Page 31	MSGIMPC
CMEMIE_IM_FEXT	4c	Page 28	.IMP
CMEMIE_IM_IEXT	4c	Page 28	.PMI
CMEMIE_IM_PEXT	4c	Page 28	.IPD
CMEMIE_IM_EEXT	4c	Page 29	.ERR
CMEMIE_IM_MFIL	1c	Page 29	M
CMEMIE_IM_NTYP	1c	Page 29	S
CMEMIE_IM_DELF	1c	Page 29	N
CMEMIE_IM_MFLD	8c	Page 28	MSGCMNT

Table 9. The Import Process Profile Structure (continued)

Tag	Type	Explanation	Default
CMEMIE_IM_MQQM	48c	Page 30	MIEQM
CMEMIE_IM_MQRQ	48c	Page 30	ACKIMP
CMEMIE_IM_MQSQ	48c	Page 30	MSGIMP
CMEMIE_IM_MQFQ	48c	Page 30	REFIMP
CMEMIE_IM_MQCQ	48c	Page 30	MSGIMPC
CMEMIE_IM_MQVS	1c	Page 31	N
CMEMIE_IM_MEXT	4c	Page 30	.MQI
CMEMIE_IM_NEXT	4c	Page 31	.NAK
CMEMIE_IM_PENC	1c	Page 31	0

Sample Import Process Profiles

The following sample profiles show you examples for different import processes.

Import Process 1

```
CMEMIE_IM_LOGL = 1
CMEMIE_IM_IDEN = IMPORT1
CMEMIE_IM_PATH = C:\M2IMPEXP\IMPORT1
CMEMIE_IM_POLL = 10
CMEMIE_IM_CTRL = MSGIMPC1
CMEMIE_IM_DELF = N
CMEMIE_IM_FEXT = .OUT
CMEMIE_IM_IEXT = .TMP
CMEMIE_IM_PEXT = .IMP
CMEMIE_IM_EEXT = .ERR
CMEMIE_IM_MFIL = A
CMEMIE_IM_NTYP = S
```

Import Process 2

```
CMEMIE_IM_LOGL = 2
CMEMIE_IM_IDEN = IMPORT2
CMEMIE_IM_PASS = IMPORT2
CMEMIE_IM_PATH = C:\M2IMPEXP\IMPORT2
CMEMIE_IM_POLL = 35
CMEMIE_IM_CTRL = MSGIMPC2
CMEMIE_IM_DELF = Y
CMEMIE_IM_FEXT = .OUT
CMEMIE_IM_IEXT = .TMP
CMEMIE_IM_PEXT = .IMP
CMEMIE_IM_EEXT = .ERR
CMEMIE_IM_MFIL = M
CMEMIE_IM_NTYP = O
```

Tags and Values for the Export Process Profile Structure

The following table lists the:

1. Tags of the export process profile
2. Data types of the tag values
3. Default tag values that are taken if the corresponding tags are missing or if the tag values are not valid

Note that the type description is divided into numbers and characters. The number specifies the maximum length of the tag values. The character **c** specifies that the value must be a character string. The character **n** specifies that the value must be an integer number.

The tags in the process profile can be in arbitrary order.

Table 10. The Export Process Profile Structure

Tag	Type	Explanation Reference	Default
CMEMIE_EX_LOGL	1n	Page 27	1
CMEMIE_EX_MAXL	7n	Page 27	100
CMEMIE_EX_SAVC	256c	Page 28	
CMEMIE_EX_APPL	8c	Page 28	value of CMEMIE_EX_IDEN
CMEMIE_EX_IDEN	8c	Page 30	EXPORT
CMEMIE_EX_PASS	8c	Page 31	
CMEMIE_EX_CPRF	80c	Page 31	exp_con.prf
CMEMIE_EX_PATH	80c	Page 28	
CMEMIE_EX_CTRL	8c	Page 31	MSGEXPC
CMEMIE_EX_QUEU	8c	Page 31	MSGEXP
CMEMIE_EX_ALRM	8c	Page 31	ALRMEXP
CMEMIE_EX_IEXT	4c	Page 29	.PXE
CMEMIE_EX_FEXT	4c	Page 29	.EXP
CMEMIE_EX_LONG	1c	Page 29	N
CMEMIE_EX_MFIL	1c	Page 29	M
CMEMIE_EX_TOUT	4n	Page 29	30
CMEMIE_EX_MMIF	4n	Page 29	5
CMEMIE_EX_MFLD	8c	Page 28	MSGCMNT
CMEMIE_EX_MQQM	48c	Page 30	MIEQM
CMEMIE_EX_MQRQ	48c	Page 30	MSGEXP
CMEMIE_EX_MQCQ	48c	Page 30	MSGEXPC
CMEMIE_EX_MQSQ	48c	Page 30	ACKEXP
CMEMIE_EX_MQVS	1c	Page 31	N
CMEMIE_EX_PENC	1c	Page 31	0

Sample Export Process Profiles

The following sample process profile shows you an example for an export process.

Export Process

```
CMEMIE_EX_LOGL = 1
CMEMIE_EX_IDEN = EXPORT1
CMEMIE_EX_PATH = C:\M2IMPEXP\EXPORT
CMEMIE_EX_QUEU = MSGEXP
CMEMIE_EX_ALRM = ALRMEXP
CMEMIE_EX_CTRL = MSGEXPC1
CMEMIE_EX_MMIF = 10
```

```
CMEMIE_EX_TOUT = 60
CMEMIE_EX_FEXT = .OUT
CMEMIE_EX_IEXT = .TMP
CMEMIE_EX_MFIL = A
CMEMIE_EX_SAVC = cmd /c ren %1 %2.sic
```

Appendix B. Configuration of an MERVA ESA MQSeries NT Scenario

The following examples show you different configuration files that you need to connect MIE running on Windows NT (MSEPCXTZ) to MERVA ESA running in a CICS/ESA® environment (MERCICS1), as described in "MERVA ESA MQSeries NT Scenario" on page 5. MERVA ESA uses MERVA-MQI Attachment for the message exchange via MQSeries. The parameters that must have the same value as the parameters of other configuration files are shown in bold. For detailed information about CICS/ESA, refer to the *MERVA ESA Customization Guide*.

MIE Profiles

The following example shows you the MIE import profile:

```
CMEMIE_IM_QTYP     =       MQI                  # queue system
CMEMIE_IM_MQQM     =       MSEPCXTZQM1          # MQI queue manager name
CMEMIE_IM_MQRQ     =       MIEIMPR              # MQI receive queue
CMEMIE_IM_MQSQ     =       MIEIMPS              # MQI send queue
CMEMIE_IM_MQCQ     =       MIEIMPC              # MQI control queue
CMEMIE_IM_MQFQ     =       MIEIMPF              # MQI reference queue
CMEMIE_IM_MQVS     =       N                    # remote system type
CMEMIE_IM_LOGL     =       1                    # log level
CMEMIE_IM_MAXL     =       10000                # max log file length (in k-bytes)
CMEMIE_IM_SAVC     =                            # command to be executed when MAXL
                                                  reached
CMEMIE_IM_APPL     =       IMPORT               # application name
CMEMIE_IM_PATH     =       e:\mie_mqi\msg       # import files directory
CMEMIE_IM_POLL     =       5                    # polling timer
CMEMIE_IM_FEXT     =       .IMP                 # initial import file extension
CMEMIE_IM_IEXT     =       .PMI                 # intermediate import file extension
CMEMIE_IM_PEXT     =       .IPD                 # imported file extension
CMEMIE_IM_EEXT     =       .ERR                 # erroneous import file extension
CMEMIE_IM_MEXT     =       .MQI                 # waiting for ack. extension
CMEMIE_IM_NEXT     =       .NAK                 # NAK messages extension
CMEMIE_IM_MFIL     =       A                    # message file format
CMEMIE_IM_NTYP     =       S                    # MERVA message header type
CMEMIE_IM_DELF     =       N                    # delete file after import
CMEMIE_IM_MFLD     =       MSGCMNT              # message routing field
```

The following example shows you the MIE export profile:

```
CMEMIE_EX_QTYP     =       MQI                  # queue system
CMEMIE_EX_MQQM     =       MSEPCXTZQM1          # MQI queue manager name
CMEMIE_EX_MQRQ     =       MIEEXPR              # MQI receive queue
CMEMIE_EX_MQSQ     =       MIEEXPS              # MQI send queue
CMEMIE_EX_MQCQ     =       MIEEXPC              # MQI cntrol queue
CMEMIE_EX_MQFQ     =       MIEEXPF              # MQI reference queue
CMEMIE_EX_MQVS     =       N                    # remote system type
CMEMIE_EX_LOGL     =       1                    # log level
CMEMIE_EX_MAXL     =       10000                # max log file length (in k-bytes)
CMEMIE_EX_SAVC     =                            # command to be executed when MAXL
                                                  reached
CMEMIE_EX_APPL     =       EXPORT               # application name
CMEMIE_EX_PATH     =       e:\mie_mqi\msg       # export files directory
CMEMIE_EX_FEXT     =       .EXP                 # final export file extension
CMEMIE_EX_IEXT     =       .PXE                 # intermediate final export file
                                                  extension
CMEMIE_EX_LONG     =       N                    # long name format for exported files
CMEMIE_EX_TOUT     =       5                    # time to wait before closing file
```

```
CMEMIE_EX_MMIF     =       5                    # max. number of messages in a file
CMEMIE_EX_MFIL     =       A                    # message file format
CMEMIE_EX_MFLD     =       MSGCMNT              # message routing field
```

MQSeries Configuration File on Windows NT

The following example shows a sample MQSeries NT configuration file with the name CMECMMQI.CFG. This file is located in the directory into which MIE was installed. To use it to configure a queue manager with the name MSEPCXTZQM1:

1. Define the MQSeries queue manager with the name MSEPCXTZQM1.
2. Start MSEPCXTZQM1.
3. Enter the command:

   ```
   RUNMQSC < amief_path\CMECMMQI.CFG
   ```

   where `amief_path` represents the path to the directory into which MIE was installed. This command starts the utility that you can use to configure MSEPCXTZQM1 using the file CMECMMQI.CFG.

```
*------------------------------------------------------------------------------*
*  Filename cmecmmqi.cfg
*  ---------------------
*  This is a sample MQSeries NT Configuration file
*
*  REVISION LEVEL: 1.0
*------------------------------------------------------------------------------*
DELETE QLOCAL(DLQ2)          PURGE
DELETE QLOCAL(DXQ2)          PURGE

*----------------*
* MIE channels   *
*----------------*

STOP   CHANNEL(MERCICS1.TO.MSEPCXTZ)    MODE(FORCE)
DELETE CHANNEL(MERCICS1.TO.MSEPCXTZ)
STOP   CHANNEL(MSEPCXTZ.TO.MERCICS1)    MODE(FORCE)
DELETE CHANNEL(MSEPCXTZ.TO.MERCICS1)
*
DELETE QLOCAL(CSQ1)           PURGE
DELETE QMODEL(STOPQ)

*----------------*
* Export queues  *
*----------------*
DELETE QREMOTE(MIEEXPS)
DELETE QLOCAL(MIEEXPR)        PURGE
DELETE QLOCAL(MIEEXPC)        PURGE
DELETE QMODEL(EXPORT)

*----------------*
* Import queues  *
*----------------*
DELETE QREMOTE(MIEIMPS)
DELETE QLOCAL(MIEIMPR)        PURGE
DELETE QLOCAL(MIEIMPC)        PURGE
DELETE QLOCAL(MIEIMPF)        PURGE
DELETE QMODEL(IMPORT)

*----------------*
* common elements*
*----------------*
DEFINE QLOCAL(DLQ2)                                    +
       REPLACE                                         +
       DEFPRTY(0)                                      +
       DEFPSIST(YES)                                   +
```

```
       DESCR('TP Dead Letter Queue')                        +
       PUT(ENABLED)                                         +
       DEFSOPT(SHARED)                                      +
       GET(ENABLED)                                         +
       MAXDEPTH(99999)                                      +
       MAXMSGL(31000)                                       +
       MSGDLVSQ(PRIORITY)                                   +
       SHARE                                                +
       NOTRIGGER                                            +
       QDEPTHHI(0)                                          +
       QDEPTHLO(100)                                        +
       QDPHIEV(DISABLED)                                    +
       QDPLOEV(DISABLED)                                    +
       QDPMAXEV(DISABLED)                                   +
       QSVCIEV(NONE)                                        +
       SCOPE(QMGR)                                          +
       USAGE(NORMAL)

DEFINE QMODEL(STOPQ)                                        +
       REPLACE                                              +
       DEFPRTY(0)                                           +
       DEFPSIST(NO)                                         +
       DESCR('MIE Stop Queue')                              +
       PUT(ENABLED)                                         +
       DEFSOPT(SHARED)                                      +
       GET(ENABLED)                                         +
       MAXDEPTH(1)                                          +
       MAXMSGL(100)                                         +
       MSGDLVSQ(PRIORITY)                                   +
       SHARE                                                +
       NOTRIGGER                                            +
       QDEPTHHI(0)                                          +
       QDEPTHLO(100)                                        +
       QDPHIEV(DISABLED)                                    +
       QDPLOEV(DISABLED)                                    +
       QDPMAXEV(DISABLED)                                   +
       QSVCIEV(NONE)                                        +
       USAGE(NORMAL)                                        +
       DEFTYPE(TEMPDYN)

DEFINE QLOCAL(DXQ2)                                         +
       REPLACE                                              +
       DEFPRTY(0)                                           +
       DEFPSIST(YES)                                        +
       DESCR('TP Default Transmit Queue')                   +
       PUT(ENABLED)                                         +
       DEFSOPT(SHARED)                                      +
       GET(ENABLED)                                         +
       MAXDEPTH(99999)                                      +
       MAXMSGL(31000)                                       +
       MSGDLVSQ(PRIORITY)                                   +
       SHARE                                                +
       NOTRIGGER                                            +
       QDEPTHHI(0)                                          +
       QDEPTHLO(100)                                        +
       QDPHIEV(DISABLED)                                    +
       QDPLOEV(DISABLED)                                    +
       QDPMAXEV(DISABLED)                                   +
       QSVCIEV(NONE)                                        +
       SCOPE(QMGR)                                          +
       USAGE(XMITQ)

*------------------*
* Channels for MIE *
*------------------*

DEFINE CHANNEL(MERCICS1.TO.MSEPCXTZ)                        +
```

```
       CHLTYPE(RCVR)                                          +
       TRPTYPE(TCP)                                           +
       DESCR('MIE Receive Channel from ESA')                  +
       REPLACE                                                +
       MAXMSGL(32000)                                         +
       SEQWRAP(999999999)                                     +
       MRRTY(3)                                               +
       MRTMR(30)

DEFINE CHANNEL(MSEPCXTZ.TO.MERCICS1)                          +
       CHLTYPE(SDR)                                           +
       CONNAME('9.164.182.254(1414)')                         +
       TRPTYPE(TCP)                                           +
       XMITQ(CSQ1)                                            +
       CONVERT(NO)                                            +
       DESCR('MIE Send Channel to ESA')                       +
       REPLACE                                                +
       DISCINT(0)                                             +
       LONGRTY(999999999)                                     +
       LONGTMR(30)                                            +
       MAXMSGL(32000)                                         +
       SEQWRAP(999999999)                                     +
       SHORTRTY(3)                                            +
       SHORTTMR(3)


*--------------------------------------*
* Xmit queue used for Export AND Import *
*--------------------------------------*

*     this local Xmit q shall be named as the remote queue manager !
DEFINE QLOCAL(CSQ1)                                            +
       REPLACE                                                 +
       DEFPRTY(0)                                              +
       DEFPSIST(YES)                                           +
       DESCR('MIE Xmit Queue to ESA')                          +
       PUT(ENABLED)                                            +
       DEFSOPT(SHARED)                                         +
       GET(ENABLED)                                            +
       MAXDEPTH(99999)                                         +
       MAXMSGL(31000)                                          +
       MSGDLVSQ(PRIORITY)                                      +
       SHARE                                                   +
       NOTRIGGER                                               +
       QDEPTHHI(0)                                             +
       QDEPTHLO(100)                                           +
       QDPHIEV(DISABLED)                                       +
       QDPLOEV(DISABLED)                                       +
       QDPMAXEV(DISABLED)                                      +
       QSVCIEV(NONE)                                           +
       SCOPE(QMGR)                                             +
       USAGE(XMITQ)

*-------------------*
* MIE Export queues *
*-------------------*

DEFINE QLOCAL(MIEEXPR)                                         +
       REPLACE                                                 +
       DEFPRTY(0)                                              +
       DEFPSIST(YES)                                           +
       DESCR('MIE Export Receive Queue')                       +
       PUT(ENABLED)                                            +
       DEFSOPT(SHARED)                                         +
       GET(ENABLED)                                            +
       INITQ(TRIGQ)                                            +
       MAXDEPTH(99999)                                         +
```

```
       MAXMSGL(32000)                                  +
       MSGDLVSQ(PRIORITY)                              +
       SHARE                                           +
       NOTRIGGER                                       +
       QDEPTHHI(0)                                     +
       QDEPTHLO(100)                                   +
       QDPHIEV(DISABLED)                               +
       QDPLOEV(DISABLED)                               +
       QDPMAXEV(DISABLED)                              +
       QSVCIEV(NONE)                                   +
       SCOPE(QMGR)                                     +
       USAGE(NORMAL)

DEFINE QLOCAL(MIEEXPC)                                 +
       LIKE(MIEEXPR)

DEFINE QREMOTE(MIEEXPS)                                +
       REPLACE                                         +
       DEFPRTY(0)                                      +
       DEFPSIST(YES)                                   +
       DESCR('MIE Export Send Queue')                  +
       PUT(ENABLED)                                    +
       RNAME(MIEEXPRCVQ)                               +
       RQMNAME(CSQ1)                                   +
       XMITQ(CSQ1)                                     +
       SCOPE(QMGR)

DEFINE QMODEL(EXPORT)                                  +
       LIKE(STOPQ)

*-------------------*
* MIE Import queues *
*-------------------*

DEFINE QLOCAL(MIEIMPR)                                 +
       REPLACE                                         +
       DEFPRTY(0)                                      +
       DEFPSIST(YES)                                   +
       DESCR('MIE Import Receive Queue')               +
       PUT(ENABLED)                                    +
       DEFSOPT(SHARED)                                 +
       GET(ENABLED)                                    +
       INITQ(TRIGQ)                                    +
       MAXDEPTH(99999)                                 +
       MAXMSGL(32000)                                  +
       MSGDLVSQ(PRIORITY)                              +
       SHARE                                           +
       NOTRIGGER                                       +
       QDEPTHHI(0)                                     +
       QDEPTHLO(100)                                   +
       QDPHIEV(DISABLED)                               +
       QDPLOEV(DISABLED)                               +
       QDPMAXEV(DISABLED)                              +
       QSVCIEV(NONE)                                   +
       SCOPE(QMGR)                                     +
       USAGE(NORMAL)

DEFINE QLOCAL(MIEIMPC)                                 +
       LIKE(MIEIMPR)

DEFINE QREMOTE(MIEIMPS)                                +
       REPLACE                                         +
       DEFPRTY(0)                                      +
       DEFPSIST(YES)                                   +
       DESCR('MIE Import Send Queue')                  +
       PUT(ENABLED)                                    +
       RNAME(MIEIMPRCVQ)                               +
```

```
       RQMNAME(CSQ1)                                          +
       XMITQ(CSQ1)                                            +
       SCOPE(QMGR)

DEFINE QLOCAL(MIEIMPF)                                        +
       REPLACE                                                +
       DEFPRTY(0)                                             +
       DEFPSIST(YES)                                          +
       DESCR('MIE Import Reference Queue')                    +
       PUT(ENABLED)                                           +
       DEFSOPT(SHARED)                                        +
       GET(ENABLED)                                           +
       MAXDEPTH(99999)                                        +
       MAXMSGL(31000)                                         +
       MSGDLVSQ(PRIORITY)                                     +
       SHARE                                                  +
       NOTRIGGER                                              +
       QDEPTHHI(0)                                            +
       QDEPTHLO(100)                                          +
       QDPHIEV(DISABLED)                                      +
       QDPLOEV(DISABLED)                                      +
       QDPMAXEV(DISABLED)                                     +
       QSVCIEV(NONE)                                          +
       SCOPE(QMGR)                                            +
       USAGE(NORMAL)

DEFINE QMODEL(IMPORT)                                         +
       LIKE(STOPQ)
```

MQSeries Configuration File on MERVA ESA

The following example shows you an MQSeries configuration file on MERVA ESA.

```
//*MAIN CLASS=CICS
//COMMAND EXEC PGM=CSQUTIL,PARM='CSQ1'
//STEPLIB  DD DSN=SYS1.MQM.SCSQANLE,DISP=SHR
//         DD DSN=SYS1.MQM.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
COMMAND DDNAME(CMDINP)
/*
//CMDINP   DD *
************************************************
*This Job generates all queues for MIE with a PS/2*
************************************************
*
* TRIGGER PROCESS FOR MIE PS/2
*
DELETE PROCESS(DSL.MERVA33.CICS410.MIETRIGGER)
DEFINE PROCESS(DSL.MERVA33.CICS410.MIETRIGGER)                +
       DESCR('TRIGGER PROCESS FOR MIE')                       +
       APPLICID('CSQX START')                                 +
       APPLTYPE(MVS)                                          +
       USERDATA('MERVA33.CHANNEL.MIE')                        +
       ENVRDATA(' ')                                          +
       REPLACE
*
*********************************************
* Local Definition of the Remote Receive Queue *
*********************************************
*
DELETE QLOCAL(MSEPCXTZQM1)
DEFINE QLOCAL(MSEPCXTZQM1)                                    +
       LIKE(QMNAMEB.XMIT.QUEUE)                               +
       PROCESS(DSL.MERVA33.CICS410.MIETRIGGER)                +
       DESCR('MUST HAVE THE SAME NAME AS THE REMOTE QMGR')    +
       REPLACE
```

```
*
* REMOTE RECEIVE QUEUE IMPORT
*
DELETE QREMOTE(MIEIMPSNDQ)
DEFINE QREMOTE(MIEIMPSNDQ)                                          +
       DESCR('Queue for Accessing RRCVQ on RQMGR')                  +
       RNAME(MIEIMPR)                                               +
       RQMNAME(MSEPCXTZQM1)                                         +
       XMITQ(MSEPCXTZQM1)                                           +
       PUT(ENABLED)                                                 +
       DEFPSIST(YES)                                                +
       DEFPRTY(9)                                                   +
       REPLACE

*
* REMOTE RECEIVE QUEUE EXPORT
*
DELETE QREMOTE(MIEEXPSNDQ)
DEFINE QREMOTE(MIEEXPSNDQ)                                          +
       DESCR('Queue for Accessing RRCVQ on RQMGR')                  +
       RNAME(MIEEXPR)                                               +
       RQMNAME(MSEPCXTZQM1)                                         +
       XMITQ(MSEPCXTZQM1)                                           +
       PUT(ENABLED)                                                 +
       DEFPSIST(YES)                                                +
       DEFPRTY(9)                                                   +
       REPLACE

*
* LOCAL RECEIVE QUEUE EXPORT
*
DELETE QLOCAL(MIEEXPRCVQ)
DEFINE QLOCAL(MIEEXPRCVQ)                                           +
       LIKE(DSL.MQI.RECEIVE)                                        +
       PROCESS(DSL.MERVA33.CICS410.MIETRIGGER)                      +
       REPLACE
*
* LOCAL RECEIVE QUEUE IMPORT
*
DELETE QLOCAL(MIEIMPRCVQ)
DEFINE QLOCAL(MIEIMPRCVQ)                                           +
       LIKE(DSL.MQI.RECEIVE)                                        +
       PROCESS(DSL.MERVA33.CICS410.MIETRIGGER)                      +
       REPLACE

****************************************
* Sender Channel for a TCP/IP Connection *
****************************************
*
* Send channel to MIE PS/2 System
*
DELETE CHANNEL(MERCICS1.TO.MSEPCXTZ)
DEFINE CHANNEL(MERCICS1.TO.MSEPCXTZ)                                +
       DESCR('Channel for Sending Messages to RQMGR')               +
       CHLTYPE(SDR)                                                 +
       CONNAME('9.164.167.23')                                      +
       TRPTYPE(TCP)                                                 +
       XMITQ(MSEPCXTZQM1)                                           +
       MCAUSER(' ')                                                 +
       BATCHSZ(50)                                                  +
       DISCINT(6000)                                                +
       SHORTRTY(10)                                                 +
       SHORTTMR(60)                                                 +
       LONGRTY(999999999)                                           +
       LONGTMR(1200)                                                +
       SCYEXIT(' ')                                                 +
       SCYDATA(' ')                                                 +
```

```
       MSGEXIT('DSLKCM1M')                                          +
       MSGDATA('OS2')                                               +
       SENDEXIT(' ')                                                +
       SENDDATA(' ')                                                +
       RCVEXIT(' ')                                                 +
       RCVDATA(' ')                                                 +
       SEQWRAP(999999999)                                           +
       MAXMSGL(4194304)                                             +
       REPLACE

******************************************
* Receiver Channel for a TCP/IP Connection *
******************************************
*
* Receive channel from PS/2 MIE
*
DELETE CHANNEL(MSEPCXTZ.TO.MERCICS1)
DEFINE CHANNEL(MSEPCXTZ.TO.MERCICS1)                                +
       DESCR('Channel for Receiving Messages from RQMGR')           +
       CHLTYPE(RCVR)                                                +
       TRPTYPE(TCP)                                                 +
       MCAUSER(' ')                                                 +
       BATCHSZ(50)                                                  +
       SCYEXIT(' ')                                                 +
       SCYDATA(' ')                                                 +
       MSGEXIT('DSLKCM1M')                                          +
       MSGDATA(' ')                                                 +
       SENDEXIT(' ')                                                +
       SENDDATA(' ')                                                +
       RCVEXIT(' ')                                                 +
       RCVDATA(' ')                                                 +
       PUTAUT(DEF)                                                  +
       SEQWRAP(999999999)                                           +
       MAXMSGL(4194304)                                             +
       REPLACE
/*
```

MQ Attachment Configuration on MERVA ESA

The following example shows an additional configuration on MERVA ESA for MQ Attachment.

```
* MACRO DSLFNTTC
*********************************************************************
*        MERVA-MQI ATTACHMENT CUSTOMIZATION FOR MIE
*********************************************************************
*
*              MERVA-TO-MQI SEND QUEUE 5
         AIF   (&IMS).IMQI5
         DSLFNT NAME=DSLMRSQ5,QUEUE=YES,THRESH=0,                    *
               TRAN=DSLS,MQI=YES,                                    *
               DESCR='Send queue 5 for MERCICS1 TO MSEPCXTZ'
         AGO   .CMQI5
.IMQI5   ANOP
         DSLFNT NAME=DSLMRSQ5,QUEUE=YES,THRESH=0,                    *
               TRAN=DSLS,MQI=YES,MSGLIM=10,                          *
               DESCR='Send queue 5 for MERCICS1 TO MSEPCXTZ'
.CMQI5   ANOP
 *
*/*LIB  MER  DSLKPSAM ASSEMBLE 97/09/10 12:00:00 MERVAESA *    *     */
*********************************************************************
*        MERVA-MQI ATTACHMENT CUSTOMIZATION FOR MIE
*********************************************************************
        DSLKPROC TYPE=INITIAL
*
*
* ROUTE MESSAGES FROM MERCICS1 TO MIE EXPORT
```

```
*
        DSLKPROC TYPE=SEND,                                          *
              NAME=SPROC5,                                           *
              EXIT=8001,                                             *
              ACKWQ=DSLMRAWQ,                                        *
              COAWQ=DSLMRCOA,                                        *
              CODWQ=DSLMRCOD,                                        *
              ALLSNDQ=((DSLMRSQ5,MIEEXPSNDQ)),                       *
              MQICTLQ=DSL.MERVA33.CICS410.MQI.CONTROL,               *
              MRVCTLQ=(DSLMRCQS,CONTINUE),                           *
              MRVSTAQ=DSLMRSTS,                                      *
              MQFMT=BLANK,                                           *
              REPLYTQ=DSL.MERVA33.CICS410.MQI.REPLY_TO_Q,            *
              OPMSDM=SUBSET
*
*
        DSLKPROC TYPE=RECEIVE,                                       *
              NAME=RPROC1,                                           *
              MQIRCVQ=(DSL.MERVA33.CICS410.MQI.REPLY_TO_Q),          *
              MRVCTLQ=DSLMRCQR,                                      *
              MRVSTAQ=DSLMRSTR,                                      *
              JRNRPLY=YES,                                           *
              EXIT=8002,                                             *
              OPMSDM=SUBSET
*
*
        DSLKPROC TYPE=RECEIVE,                                       *
              NAME=RPROC5,                                           *
              MQIRCVQ=(MIEIMPRCVQ),                                  *
              MRVCTLQ=DSLMRCQ1,                                      *
              MRVSTAQ=DSLMRSTR,                                      *
              OPMSDM=SUBSET
*
*
        DSLKPROC TYPE=RECEIVE,                                       *
              NAME=RPROC6,                                           *
              MQIRCVQ=(MIEEXPRCVQ),                                  *
              MRVCTLQ=DSLMRCQ1,                                      *
              MRVSTAQ=DSLMRSTR,                                      *
              OPMSDM=SUBSET
*
*
        DSLKPROC TYPE=FINAL
        END
```

Overview of the Configuration Files

The following figures reflect the example configuration files shown in this appendix.

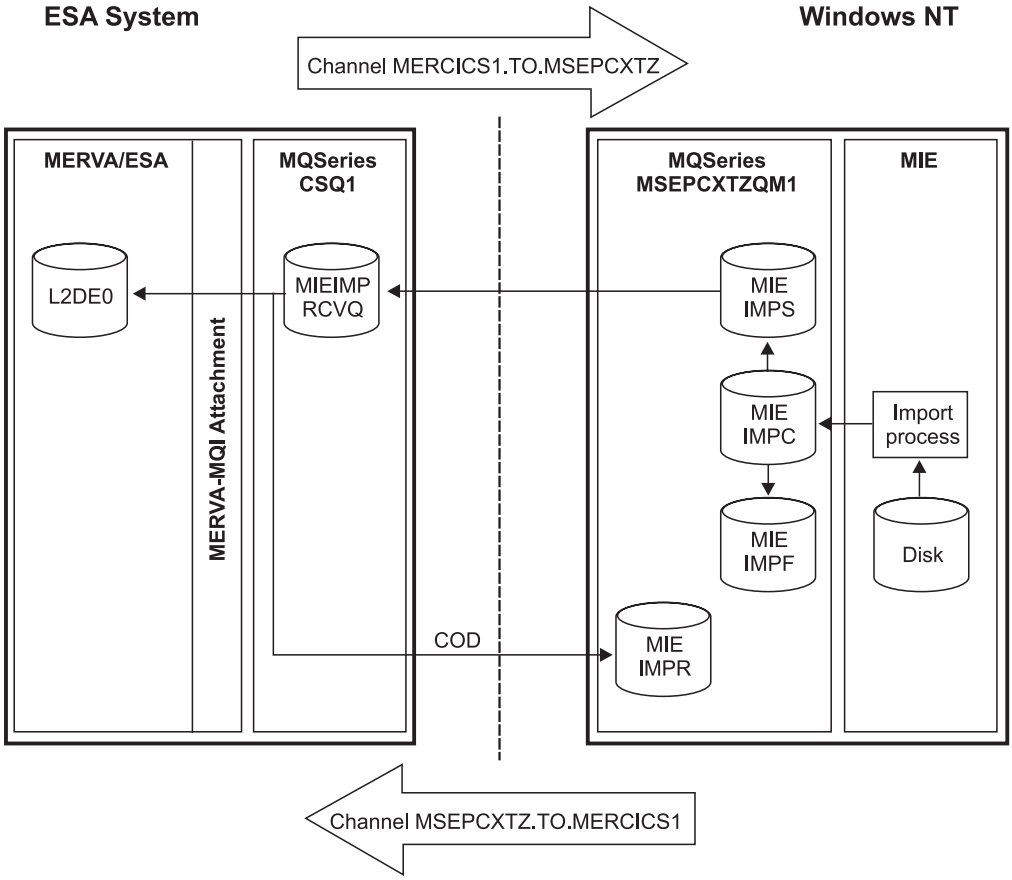

Figure 13. The MIE Import Process

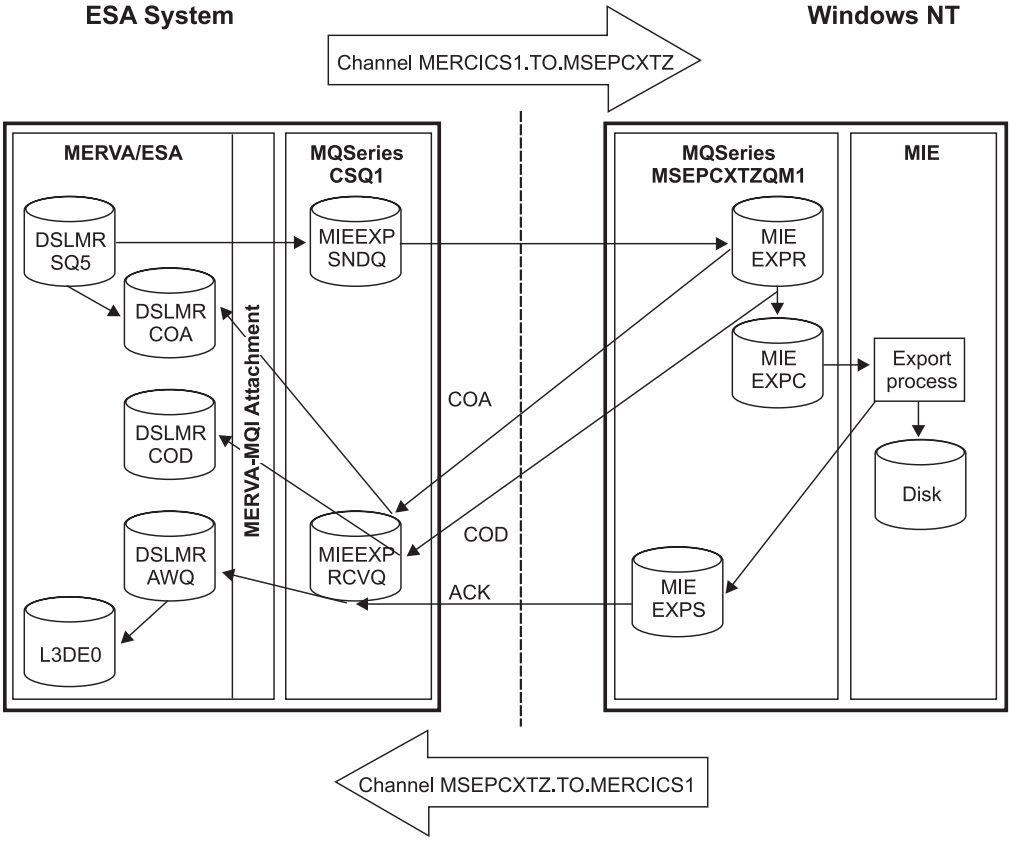

Figure 14. The MIE Export Process

Appendix C. Structures of Import and Export Message Files

MIE supports the following message formats:
- MERVA format
- AccordWorkstation format

MERVA Format

MERVA messages are stored in a binary file. All bytes that are found in the file are taken as they are. The carriage return character, the line feed character, and 0x1A sequences that some editors might append can disturb processing of the message file.

A 4-byte field that contains an unsigned long integer value precedes each message. This value represents the length of the message file. The following rules apply for the bytes:

- The first byte is the least important byte.
- The second byte is the most important byte.
- The third and fourth byte must always contain 0x00 values.

The total length of the record is the length of the message part that is added to the four bytes that precede the message part.

The following table shows you the definition of the general MERVA message format for messages of the SWIFT and OWN net types:

Table 11. Format of MERVA Message Files of Net Types SWIFT and OWN

Byte	Contents
1	Length of the first message
2	
3	
4	
5	Message part of first message (SWIFT type or OWN type)
6	
...	
n	
n+1	Length of the second message
n+2	
n+3	
n+4	
n+5	Message part of second message (SWIFT type or OWN type)
n+6	
...	
n+m	
n+m+1	Further MERVA formatted messages
...	

AccordWorkstation Format

AccordWorkstation messages are stored in a binary file. All bytes that are found in the file are taken as they are. Some editors might append the carriage return character, the line feed character, and 0x1A sequences. These characters and sequences do not disturb processing of the message file if they are located outside the file area. They are then located outside the character labels start-of-heading and end-of-text.

Each message is preceded by the start-of-heading byte <SOH>. Also, each message is followed by the end-of-text byte <ETX>. ASCII 0x01 codes the label <SOH>, and ASCII 0x03 codes the label <ETX>.

The following table shows you the definition of the general AccordWorkstation message format:

Table 12. Format of AccordWorkstation Message File

Byte	Contents
1	<SOH> (ASCII 0x01)
2	Message part of first message (SWIFT type)
3	
...	
n-1	
n	<ETX> (ASCII 0x03)
n+1	This information is not considered.
...	
n+m	<SOH> (ASCII 0x01)
n+m+1	Message part of second message (SWIFT type)
...	
...	
n+m+p	<ETX> (ASCII 0x03)
...	This information is not considered.
...	
...	Further AccordWorkstation formatted messages
...	

Appendix D. Examples of MERVA Routing Conditions

This appendix gives examples of routing conditions for the following scenarios:
- The MERVA Workstation scenario, which is described on page 3
- The MERVA ESA front-end scenario, which is described on page 4

Routing Conditions for the MERVA Workstation Scenario

The following sample routing conditions ensure that:

- Imported messages are routed directly from the MIE import control queue to the MERVA SWIFT Link send queue
- Received SWIFT messages, for example of type 998 of the MERVA Workstation SWIFT Link receive queue SLINCMS2, are routed directly to the MIE export queue

Table 13. Sample Routing Conditions for the MERVA Workstation Scenario

Messages in queue ...	**Are sent by default to ...**	**Except when ...**	**Then the messages are routed to ...**	**Further routing ...**
MSGIMPC	MSGIMPC	MIEROUTE = MIE_LOAD	SLRNM02	None
MSGEXP	MSGEXP	MIEROUTE = MIE_ERRO	MBERROR	None
		MIEROUTE = MIE_TOUL	MSGEXPC	
MSGEXPC	MSGEXPC	MIEROUTE = MIE_UNLD	MBDELETE	None
		MIEROUTE = MIE_BACK	MSGEXP	
SLINCMS2	SLINCMS2	MSGTYPE = MTYP_998	MSGEXP	None

Routing Conditions for the MERVA ESA Front-End Scenario

The following sample routing conditions ensure that:

- Imported messages are routed directly from the MIE import control queue to the MERVA Link send queue MLSNDNRM
- Received SWIFT messages, for example of type 998 of the MERVA Link receive queue MLRECEIV, are routed directly to the MIE export queue

Table 14. Sample Routing Conditions for the MERVA ESA Front-End Scenario

Messages in queue ...	**Are sent by default to ...**	**Except when ...**	**Then the messages are sent to ...**	**Further routing ...**
MSGIMPC	MSGIMPC	MIEROUTE = MIE_LOAD	MLSNDNRM	None
MSGEXP	MSGEXP	MIEROUTE = MIE_ERRO	MBMERROR	None
		MIEROUTE = MIE_TOUL	MSGEXPC	
MSGEXPC	MSGEXPC	MIEROUTE = MIE_UNLD	MBDELETE	None
		MIEROUTE = MIE_BACK	MSGEXP	
MLRECEIV	MLRECEIV	MSGNETID = SWIFTNET and MSGTYPE = MTYP_998	MSGEXP	None

Appendix E. Password Encryption Utility

The password encryption utility provided by MIE lets you avoid having to specify your password each time a process is started while still maintaining the secrecy of your password. The password encryption utility:

1. Checks whether the password is set in the corresponding process profile and, if it is not, prompts you for it
2. Encrypts the password
3. Writes the encrypted password back to the profile
4. Adds to the profile the parameter CMEMIE_EX_PENC (for an export profile) or CMEMIE_IM_PENC (for an import profile) to indicate that the password is encrypted

To invoke the password encryption utility, enter

cmenmpec *profilename* **[-n]**

where:

profilename
The fully-qualified filename of the process profile, for example c:\merva\mie\cmecmexp.prf.

-n This parameter is optional. If specified, you are prompted for both the user ID and the password regardless of whether values for the user ID and the password are already set in the specified profile. The values you enter are written to the profile (the password is first encrypted). If the profile already contains other values, these are overwritten.

If you invoke the utility without the **-n** parameter, the default values for the user ID and the password keywords are used (CMEMIE_EX_IDEN and CMEMIE_EX_PASS for an export profile; CMEMIE_IM_IDEN and CMEMIE_IM_PASS for an import profile). The utility checks if both values are set, prompts you for values if they are not, and stores the encrypted password in the profile.

Appendix F. Service Installation and Administration Utility

The service installation and administration utility (cmenmadm.exe) is an application program that helps you to install and configure a Windows NT service. To use it, enter one of the following commands in a Command Prompt window:

Table 15.

To install a service	**cmenmadm.exe -is** *service_name* **-se** *service_exe_file* **-sp** *additional_service_parameters* [**-sm** *message_catalog_dll_file*] [**-sdn** *service_display_name*] [**-sd** *service_dependencies*] [**-sua** *user_account* **-sup** *user_password*] [**-ds**|**-as**]
To update the parameters of an installed service	**cmenmadm.exe -us** *service_name* [**-se** *service_exe_file*] [**-sp** *additional_service_parameters*] [**-sm** *message_catalog_dll_file*] [**-sdn** *service_display_name*] [**-sd** *service_dependencies*] [**-sua** *user_account* **-sup** *user_password*] [**-ds**|**-as**]
To start a service	**cmenmadm.exe -ss** *service_name*
To query the status of a service (normal)	**cmenmadm.exe -qss** *service_name*
To query the status of a service (detailed)	**cmenmadm.exe -qs** *service_name* **Note:** The information returned by this command requires knowledge of internal Windows NT service handling.
To terminate (stop) a service	**cmenmadm.exe -ts** *service_name*
To remove (uninstall) a service	**cmenmadm.exe -rs** *service_name* **Note:** Only a service that has been terminated can be removed. After a service is removed, it must be reinstalled before it can be restarted. The removal of a service also deregisters its message catalog.
To display the command syntax	**cmenmadm.exe -h**

Parameters:

service_name
The name of the service.

service_exe_file
The name of the service executable on the local file system:

- Use **cmecmimp.exe** for an MIE import process.
- Use **cmecmexp.exe** for an MIE export process.

If you specify the file name only (that is, without a file path), the service control manager (SCM) searches all directories specified in the PATH

environment variable. If the directory containing the service is not in this path, you must specify a fully-qualified file name.

additional_service_parameters
This parameter is used to pass additional parameters to an MIE import or export service when this service is called by the SCM. For MIE, the syntax for this parameter is:

```
-sp "profile_file -s service_name [-t] [-p]"
```

where:

profile_file
The fully-qualified name of the import or export profile file.

service_name
The same service name that was specified with the -is or -us parameter.

-t Specifies that MERVA is to create a routing trace.

-p Specifies that MERVA is to log the parameters.

message_catalog_dll_file
When this parameter is specified, the service tool registers the specified message catalog by making entries in the Windows NT registry. The entries are written under the following key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\EventLog\Application\service_name\
```

For example, if you specify `-sm cmenmmes.dll`, the service installation and administration tool adds the following two entries under the key shown above:

```
EventMessageFile="cmenmmes.dll" TypesSupported=7
```

TypesSupported is set to 7 for all message types (error, warning and information).

If this parameter is omitted, a warning message is written to the console. To register the message catalog at a later time, use the update command. If the message catalog is specified without a path, the paths defined by the PATH environment variable are searched.

service_display_name
This is the name that is displayed in the Windows NT services applet. If not specified, the service name is used as the service display name. If the service display name contains blanks, it must be enclosed in quotation marks.

service_dependencies
Use this parameter to specify the names of other services that must be started before the MIE service can receive a start command from the Windows NT service control manager (SCM). If they are not running, a service for which a start service (-ss) command has been issued will wait until they are running before starting. These dependencies apply only to start, not to shutdown. Separate the names with a comma (,). If a name contains blanks, the entire list must be enclosed in quotation marks.

user_account **and** *user_password*
The Windows NT user account and password under which the service process is to run. The service tool adds the prefix **.** to the user account

string to indicate a local account before it registers the service to the SCM. The default is the Windows NT LocalSystem account.

-ds Start the service on demand; that is, the service requires an explicit start command. This is the default.

-as Windows NT is to start the service automatically.

Examples

Assume you want to do the following:

- Install an MIE service with name MIE_EXP1 and export profile cmecmexp.prf.
- Register the message catalog cmenmmes.dll, specify a display name that is different from the service name, and specify the user account and password.

The following is an example of a typical sequence of commands to do this:

```
cmenmadm.exe -is MIE_EXP1 -se cmecmexp.exe -sp "cmecmexp.prf -s MIE_EXP1"
  -sm cmenmmes.dll -sdn "MIE export 1" -sua mervauser -sup userpwd
```

Now assume that, after you install the service, you realize that you need to ensure that the MERVA instance with the name ENM_merva1 is running before the service is started. To do this, use the update service (-us) command to amend it as follows:

```
cmenmadm.exe -us MIE_EXP1 -sd ENM_merva1
```

To start the service, enter the following:

```
cmenmadm.exe -ss MIE_EXP1
cmenmadm.exe -qss MIE_EXP1
```

The -ss command starts the service. The -qss command queries the status of the service. This way, you can check whether the service started successfully.

To terminate the service, enter the following:

```
cmenmadm.exe -ts MIE_EXP1
cmenmadm.exe -qss MIE_EXP1
```

The -ts command terminates the MIE service with name MIE_EXP1. The -qss command queries the status of the service. This way, you can check whether the termination was successful.

After the service has been stopped, and if it will no longer be needed, it can be removed with the following command:

```
cmenmadm.exe -rs MIE_EXP1
```

Appendix G. Messages and Codes

The messages issued by MIE have IDs that have the format

CMEnnnne

where:

nnnn Message number

e Message classification:

- **I** Information only; no error has occured
- **W** Warning (an irregularity occured that is considered to be recoverable)
- **E** Error

CME1000I The MERVA Export Process is going to shut down.

Explanation: One of the export processes will exit.

System Action: The export process will be stopped.

User Response: None.

CME1001W The MERVA Export Process with process ID *ProcessID* has been invoked in duplicate. The second instance will be stopped immediately.

Explanation: An export process has been invoked twice with the same MERVA user or application ID, possibly with the same process profile.

System Action: The second instance will be stopped.

User Response: Check the process profile of the second export process and compare its contents with those of the first process' profile.

CME1002I The MERVA Export Process is going to start up.

Explanation: The export process will start.

System Action: The export process will be started.

User Response: None.

CME1003E The library *Library* could not be loaded.

Explanation: The indicated library could not be loaded dynamically.

System Action: The program terminates.

User Response: Verify, whether the indicated library is on the correct path or the product the product providing the library has been installed correctly.

CME1004W Intermediate message file *FileName* is recovered (EXPORT STATE 1) and then deleted. After recovery, the corresponding messages will be exported again.

Explanation: During restart, a message file that was not completely processed has been recovered. For details of the transfer state refer to the documentation.

System Action: The recovered file will be deleted.

User Response: None.

CME1005W Message with MRN = *MRN* is recovered (EXPORT STATE 2). The message is routed back with flag *Flag* to be exported again.

Explanation: During restart, a message file that was not completely processed has been recovered. The corresponding MERVA messages had been already partially exported. For details of the transfer state refer to the documentation.

System Action: The incomplete message file is deleted, and the corresponding messages are routed back to the export queue to be exported again.

User Response: None.

CME1006W Intermediate message with MRN = *MRN* is recovered (EXPORT STATE 3). The message is routed forward with flag *Flag*.

Explanation: During restart, a message that was not completely processed has been recovered. The corresponding message file had been already processed. For details of the transfer state refer to the documentation.

System Action: The remaining processing step is executed.

User Response: None.

CME1007E Critical Error during recovery detected. The process will be stopped.

Explanation: During restart, a recovery error has been occurred. More detailed information will be logged.

System Action: None.

User Response: Check the log file for corresponding entries that describe the reason for the error.

CME1008E Undefined recovery state detected: *FileNum* **intermediate files.**

Explanation: During restart, more than one intermediate message file has been detected. Usually, this error does not occur. However, specific manipulations of the import/export directories can cause this error situation.

System Action: The process will exit.

User Response: Check the import/export directory and clear up the undefined recovery situation.

CME1009W Incomplete message file *FileName* **is recovered (EXPORT STATE 2) and then deleted. After recovery, the corresponding messages will be exported again.**

Explanation: During restart, a message file that was not completely processed has been recovered. The corresponding MERVA messages were already exported partially. For details of the transfer state refer to the documentation.

System Action: The incomplete file is deleted, and the corresponding messages are routed back to the export queue to be exported again.

User Response: None.

CME1010E Critical Error during recovery: Messages could not be routed from the control queue.

Explanation: An internal error of a MERVA router might prevent the process from routing the messages from the control queue.

System Action: The process will exit.

User Response: Check the MERVA log files to find out the reason for the error.

CME1015I Start loading customization data of the MERVA Export Process from customization profile *ProfileName***.**

Explanation: The process profile will be read from the file *ProfileName*.

System Action: None.

User Response: None.

CME1016I Customization data of the MERVA Export Process *ProcessID* **was loaded successfully.**

Explanation: The process profile of the process with the MERVA application name *ProcessID* has been read.

System Action: None.

User Response: None.

CME1018W Stopping the MERVA Import/Export *ProcessID* **has been initiated.**

Explanation: The stop process has cleared the stop semaphore of the import/export process *ProcessID*.

System Action: The process *ProcessID* will stop.

User Response: None.

CME1019E Error during moving a file from *SourceFile* **to** *DestFile* **in Function** *"FuncName"* **(rc =** *ReturnCode***).**

Explanation: A critical error occurred during renaming the message file *SourceFile* in *DestFile*. The import/export subfunction is reported by *FuncName*; the system error code is *ReturnCode*.

System Action: The system tries to perform the operation again, or the process stops.

User Response: Check whether the directories contain files with the same file name as the target's one. Check the system error code to find out the reason for the error. Stop all import/export processes, clear up the directories, and start again.

CME1020E Writing to file *FileName* **failed (***BytesToWrite/BytesWritten***).**

Explanation: A critical error occurred during writing to the message file *FileName*. *BytesToWrite* reports the number of bytes to write, *BytesWritten* the number of bytes written.

System Action: The system tries to perform the operation again, or the process stops.

User Response: Check the directories whether enough space is available, or whether there is another reason for the error.

CME1021E Unexpected termination of a process (reason = *Reason*).

Explanation: A critical error caused that the process is abended. *Reason* reports the reason code that is returned by the system.

System Action: The process will exit.

User Response: Find the reason for the abend of the process and remove the reported problem.

CME1022E Insufficient memory to process the function.

Explanation: The memory is not sufficient to allocate dynamic memory space.

System Action: The process will exit.

User Response: Find the reason for insufficient memory. Then extend the RAM, or avoid transferring large messages.

CME1023E Critical file open error - (*Reason*, *FileName*).

Explanation: A critical error occurred during opening the file *FileName*. The reason is reported by *Reason*.

System Action: The system tries to perform the operation again, or exits.

User Response: Check the file and remove the reported problem.

CME1024E Cannot load error message file *FileName*.

Explanation: The error message file could not be opened.

System Action: None.

User Response: Check whether the MERVA Automatic Import/Export message file (cmecmmsg.cat for AIX, cmecmmsg.mtx otherwise) exists in the directory. If not, copy the file from the installation diskette into the directory. Check also the message environment variable NLSPATH (AIX or OS/2).

CME1025E Cannot load message from error message file *FileName*.

Explanation: The system cannot find the error message in the error message file.

System Action: None.

User Response: Check whether the MERVA Automatic Import/Export message file (cmecmmsg.cat for AIX, cmecmmsg.mtx otherwise) exists in the directory. If not, copy the file from the installation diskette into the directory. Check also the message environment variable NLSPATH (AIX or OS/2).

CME1026E Error in getting an error message; rc = *ReturnCode* - message number: *MsgNumber*

Explanation: The system cannot get the message text of the error message *MsgNumber*. The return code is reported by *ReturnCode*.

System Action: None.

User Response: Check whether the MERVA Automatic Import/Export message file (cmecmmsg.cat for AIX, cmecmmsg.mtx otherwise) exists in the directory. If not, copy the file from the installation diskette into the directory. Check also the message environment variable NLSPATH (AIX or OS/2).

CME1027E Critical error in a call to the operating system. (*FuncName*: *Resource*; rc = *ReturnCode*)

Explanation: A critical error occurred during an operating system's service. The calling function or the called service is reported by *FuncName*. The system resource that is addressed by the call is *Resource*. The code that is returned by the system is *ReturnCode*.

System Action: Possibly, the process is going to shut down.

User Response: Evaluate the return code and check the system resources that are touched by the call. If you cannot solve the problem, report the error to your IBM representative.

CME1028E Critical error in a call to the operating system. (Function = *FuncName*; handle = *Handle*; rc = *ReturnCode*)

Explanation: A critical error occurred during an operating system's service. The calling function or the called service is reported by *FuncName*. The system resource handle that is addressed by the call is *Handle*. The code that is returned by the system is *ReturnCode*.

System Action: Possibly, the process is going to shut down.

User Response: Evaluate the return code and check the system resources that are touched by the call. If you cannot solve the problem, report the error to your IBM representative.

CME1029E Critical error: Cannot find environment variable *EnvVar*. FunctionCall: *FuncName*, ReturnCode: *ReturnCode*

Explanation: The process could not find the environment variable denoted by *EnvVar*. The calling function or the called service is reported by *FuncName*. The system return code is *ReturnCode*.

System Action: Possibly, the process is going to shut down.

User Response: Check whether the environment variable is set within your session and correct the error that you find.

CME1030E Critical error in a call to the operating system. (*FuncName***: *Resource*, *Service*; rc = *ReturnCode*)**

Explanation: A critical error occurred during an operating system's file service. The calling function or the called service is reported by *FuncName*. The resource that is addressed by the call is *Resource*. The service is described by *Service*. The code that is returned by the system is *ReturnCode*.

System Action: Possibly, the process is going to shut down.

User Response: Evaluate the return code and check the system resources that are touched by the call. If you cannot solve the problem, report the error to your IBM representative.

CME1040W Intermediate message with MRN = *MRN* recovered (IMPORT STATE 1). The message is deleted. After recovery, the corresponding message file will be imported again.

Explanation: During restart, a message that was not completely processed has been recovered. For details of the transfer state refer to the documentation.

System Action: The recovered message will be deleted.

User Response: None.

CME1041W Intermediate file *FileName* and message with MRN = *MRN* recovered (IMPORT STATE 2). The message is routed with flag *Flag*.

Explanation: During restart, a message file that was not completely processed has been recovered. The corresponding MERVA message had been created already. For details of the transfer state refer to the documentation.

System Action: The remaining processing steps are executed.

User Response: None.

CME1042W Intermediate message file *FileName* recovered (IMPORT STATE 3). The message file is renamed to *TargetName* or deleted.

Explanation: During restart, a message file that was not completely processed has been recovered. The corresponding MERVA message had been created already. For details of the transfer state refer to the documentation.

System Action: The remaining processing step is executed.

User Response: None.

CME1049E Shared memory *SHM* not found.

Explanation: None.

System Action: None.

User Response: None.

CME1050I Process with ID *ProcessID* attached to the queuing system.

Explanation: The process was successfully attached to MERVA (resp. connected to MQSeries).

System Action: None.

User Response: None.

CME1052I Process with ID *ProcessID* detached from the queuing system.

Explanation: The process was successfully detached from MERVA (resp. disconnected from MQSeries).

System Action: None.

User Response: None.

CME1054E Request for the log protection semaphore failed.

Explanation: Uncritical logging synchronization error. The wait for logging time limit of a process has been violated.

System Action: None.

User Response: None.

CME1055E Invalid semaphore index: *SemIndex*.

Explanation: Critical semaphore error.

System Action: Possibly, the process exits.

User Response: Report this error to your IBM representative.

CME1056E Invalid semaphore handle.

Explanation: Critical semaphore error.

System Action: Possibly, the process exits.

User Response: Report this error to your IBM representative.

CME1057E **Error during wait for semaphore (rc =** *ReturnCode***).**

Explanation: An error occured during waiting for a semaphore.

System Action: None.

User Response: None.

CME1059E **Request for the MERVA attachment/detachment protection semaphore failed (***SemName***, rc =** *ReturnCode***).**

Explanation: Critical semaphore error.

System Action: Possibly, the process exits.

User Response: Start the stopped process again.

CME1061E **File** *FileName* **does not exist or is empty.**

Explanation: The opened file was empty.

System Action: Possibly, the process exits.

User Response: Stop the process and provide a valid file to the process before you start it again.

CME1062E **Error during retrieval of customization data.**

Explanation: General error during reading of the process profile.

System Action: Possibly, the process exits.

User Response: Check the log file for more detailed entries and correct the errors.

CME1064W **Tag** *TagName* **is missing in Customization File** *Profile***.** *TagName* **=** *TagValue* **is assumed.**

Explanation: The tag was missing in the process' profile.

System Action: The system takes the default value.

User Response: Check the profile and insert the missing tag with the right value.

CME1066W **Value of Tag** *TagName* **in Customization File** *Profile* **is not valid.** *TagName* **=** *TagValue* **is assumed.**

Explanation: The tag value in the process' profile is not valid.

System Action: The system takes the default value.

User Response: Check the profile and correct the invalid value.

CME1067E **Value** *TagValue* **of Tag** *TagName* **in Customization File** *Profile* **is not valid.**

Explanation: The tag value in the process' profile is not valid.

System Action: The process exits.

User Response: Check the profile, correct the invalid value and restart the process.

CME1080E **The MQSeries queue manager has not been started (rc =** *ReturnCode***).**

Explanation: The MQSeries queue manager is not yet running.

System Action: The process exits.

User Response: Start MQSeries queue manager before you start the process again.

CME1081E **The MERVA system has not been started (rc =** *ReturnCode***).**

Explanation: The MERVA system is not yet running.

System Action: The process exits.

User Response: Start MERVA before you start the process again.

CME1082E **The API user ID, or the password, or both are not defined or approved in MERVA (rc =** *ReturnCode***).**

Explanation: The attachment to MERVA failed due to a wrong user ID, or a wrong password, or both.

System Action: The process exits.

User Response: Correct the customization data before you start the process again.

CME1090E **A severe error occurred in MERVA or in an API program (***FuncName* **failed; rc =** *ReturnCode***).**

Explanation: A call to a MERVA or MQI service has failed.

System Action: Possibly, the process exits.

User Response: Check the MERVA log files to find out the reason for the reported error. If the function reported by *FuncName* is a MERVA API (resp. MQI) call, check the returned code *ReturnCode* by means of the MERVA Application Programming Guide (resp. MQSeries Application Programming Reference).

CME1091E **Path** *PathName* **for message file to load not found.**

Explanation: The message file directory path was not found. Therefore, the message file could not be loaded.

System Action: The process exits.

User Response: Check whether the message file directory has been created correctly (see the Installation section in the *MERVA Automatic Message Import/Export Facility User's Guide*). Check also whether the environment variable NLSPATH has been set to the right path. Correct the error and restart the process.

CME1100I **MERVA message of queue** *QueueName* **with MRN =** *MRN* **exported to message file** *FileName***. Message has been routed with flag** *Flag***.**

Explanation: The message has been exported successfully.

System Action: None.

User Response: None.

CME1101E **Error during export of MERVA message in queue** *QueueName* **to message file** *FileName***. Message with MRN =** *MRN* **is routed with flag** *Flag***.**

Explanation: The message has not been exported due to an error.

System Action: None.

User Response: Check further log entries to determine the error.

CME1102E **File Open Error (rc =** *ReturnCode***) during export of MERVA message in queue** *QueueName***. MRN =** *MRN***. FileName =** *FileName***.**

Explanation: The message file could not be created.

System Action: The message will not be exported.

User Response: Check the system return code and the export directory to find out the reason.

CME1103E **File Write Error (rc =** *ReturnCode***) during export of MERVA message in queue** *QueueName***. MRN =** *MRN***. FileName =** *FileName***.**

Explanation: The message contents could not be written.

System Action: The message will not be exported.

User Response: Check the system return code and the export directory to find out the reason.

CME1104E **File Close Error (rc =** *ReturnCode***) during export of MERVA messages from queue** *QueueName***. FileName =** *FileName***.**

Explanation: The generated message file could not be closed.

System Action: The messages will not be exported.

User Response: Check the system return code and the export directory to find out the reason.

CME1105E **Not enough disk space available (***AvailableSpace* **free, message size** *RequiredSpace***) to export the MERVA message from queue** *QueueName***. MRN =** *MRN***.**

Explanation: There is not enough free disk space to create the message file. At least 30000 bytes must be free after saving the message. The message file requires *RequiredSpace* bytes but only *AvailableSpace* bytes are available.

System Action: The message will not be exported.

User Response: Check the export directory for the reason and provide the required disk space.

CME1106E **Write Field Error (rc =** *ReturnCode***) during routing of the MERVA message from queue** *QueueName***. MRN =** *MRN***. FileName =** *FileName* **(already created).**

Explanation: Critical Error during writing the routing flag to the message.

System Action: The message is not routed.

User Response: Check the MERVA return code *ReturnCode*, and the MERVA log and trace files to find out the reason.

CME1107E **RoutePut error (rc =** *ReturnCode***) during routing of the MERVA message from queue** *QueueName***. MRN =** *MRN***. FileName =** *FileName* **(already created).**

Explanation: Critical error during routing of the message.

System Action: The message is not routed.

User Response: Check the MERVA return code *ReturnCode*, and the MERVA log and trace files to find out the reason.

CME1108E **File rename error during export of messages to message file** *SourceFile***/***DestFile***. (rc =** *ReturnCode***).**

Explanation: A critical error occurred during renaming of the message file *SourceFile* in *DestFile*. The system error code is *ReturnCode*.

System Action: The system retries the operation or exits.

User Response: Check whether the directories contain files with the same file name as the target's one. Check the system error code to find out the reason for the error. Stop all import/export processes, clear up the directories, and start again.

CME1109I **All messages are successfully imported from file** *FileName***.**

Explanation: All messages have been imported successfully.

System Action: None.

User Response: None.

CME1110I **Message with MRN =** *MRN* **imported from file** *FileName***.**

Explanation: The message has been imported successfully.

System Action: None.

User Response: None.

CME1111E **Message file** *FileName* **could not be opened (rc =** *ReturnCode***). Possible reason:** *Reason*

Explanation: The message file could not be opened. Possibly, it is used by another process that denies the access.

System Action: The system retries the operation or exits.

User Response: Check the system return code and the import/export directory to find out the reason.

CME1112E **Delete/Routing/WriteField error during import of a message from message file** *FileName***. MERVA Queue =** *QueueName* **(rc=***ReturnCode***).**

Explanation: Critical error during deletion or routing of the message.

System Action: The message is not deleted or routed.

User Response: Check the MERVA return code *ReturnCode*, and the MERVA log and trace files to find out the reason.

CME1113E **Rename/Delete file error during import of a message from message file** *FileName***. (rc =** *ReturnCode***).**

Explanation: A critical error occurred during renaming of the message file *FileName*. The system error code is *ReturnCode*.

System Action: The system retries the operation or exits.

User Response: Check whether the directories contain files with the same file name as the target's one. Check the system error code to find out the reason for the error. Stop all import/export processes, clear up the directories, and start again.

CME1114E **Start label not found during import of a message from message file** *FileName***.**

Explanation: The format of the message file to be imported is invalid.

System Action: The message is not imported.

User Response: Check the format of the message file.

CME1115E **Close file error during import of a message from message file** *FileName***. (rc =** *ReturnCode***).**

Explanation: The message file could not be closed.

System Action: The message will not be imported.

User Response: Check the system return code and the import directory to find out the reason.

CME1116E **End of file reached before the message could be read entirely from the message file** *FileName***. Message length =** *MsgLength***; File size =** *FileSize***.**

Explanation: The length of the message is grater than the number of unprocessed bytes left in the message file. The message file is probably corrupted.

System Action: The message will not be imported.

User Response: Check the format of the message file.

CME1117E **CreateMessage/ReadField error during import of a message from message file** *FileName***. (rc =** *ReturnCode***).**

Explanation: A critical error occurred during the creation of a message.

System Action: The message is not imported.

User Response: Check the MERVA log and trace files to find out the reason.

CME1118E **Message with length** *MsgLength* **to import from file** *FileName* **exceeds the maximum MERVA message length (***MaxLength***).**

Explanation: The message is too large.

System Action: The message is not imported.

User Response: Remove the message file from the import directory.

CME1119E **Message to import from file** *FileName* **is too short (message length:** *MsgLength* **bytes).**

Explanation: The message is too small.

System Action: The message is not imported.

User Response: Remove the message file from the import directory.

CME1120W **Erroneous import message file** *FileName1* **has been renamed to** *FileName2***.**

Explanation: The reason for the renaming has been reported with another log entry.

System Action: The message file is renamed.

User Response: None.

CME1121E **End label of message file** *FileName* **not found.**

Explanation: The format of the message file to be imported is invalid.

System Action: The message is not imported.

User Response: Check the format of the message file.

CME1122E **Import message from file** *FileName* **could not be put. MERVA import control queue** *QueueName* **not found.**

Explanation: The import control queue does not exist.

System Action: The process exits.

User Response: Check the customization and restart the process.

CME1123E **Import message from file** *FileName* **could not be put. The message header does not match the rules for** *HeaderType* **headers.**

Explanation: The message is not a valid MERVA *HeaderType* message.

System Action: The message is not imported.

User Response: Check the message format and restart the process.

CME1124E **Not enough free memory to process import message file** *FileName***.**

Explanation: The available free memory space is not sufficient to process messages of this length.

System Action: The message is not imported.

User Response: Extend your memory and restart the process.

CME1125W **Erroneous import message file** *FileName1* **has been renamed to** *FileName2***. See the file** *FileName3* **for erroneous messages.**

Explanation: The message(s) in the file *FileName1* could not be imported into the remote MERVA system. The erroneous messages are written in the file *FileName3*.

System Action: The message file is renamed.

User Response: None.

CME1126I **All messages are successfully passed to MQSeries from file** *FileName***.**

Explanation: All messages have been written successfully to MQSeries. They are then sent to the remote MERVA system.

System Action: None.

User Response: None.

CME1128E **Message with the MRN** *MRN* **has been rejected by the remote MERVA system.**

Explanation: The message with the MRN *MRN* could not be imported in the remote MERVA system and a negativ aknowledgement has been received.

System Action: None.

User Response: Correct the message in the NAK file and import it again.

CME1129I **Message with the MRN** *MRN* **has been acknowledged by the remote MERVA system.**

Explanation: The message with the MRN *MRN* has been imported in the remote MERVA system and a positiv aknowledgement has been received.

System Action: None.

User Response: None.

CME1130E **The MERVA queue** *QueueName* **does not belong to the API purpose group or the user has no right to use the named queue.**

Explanation: Either the MERVA queue specified in the customization file does not belong to the API purpose group or the user has no right to access the queue.

System Action: The process exits.

User Response: Check in your MERVA system for the queue and restart the process.

CME1131E The MERVA (resp. MQSeries) queue *QueueName* does not exist.

Explanation: The MERVA (resp. MQSeries) queue specified in the customization file does not exist.

System Action: The process exits.

User Response: Correct your customization and restart the process.

CME1132E Customized file extensions are not unique. Check and change the profile tags *Tag1 Tag2 Tag3 Tag4*.

Explanation: The customization data is ambiguous.

System Action: The process exits.

User Response: Correct your customization and restart the process.

CME1133E Customized queue names are not unique. Check and change the profile tags *Tag1 Tag2*.

Explanation: The customization data is ambiguous.

System Action: The process exits.

User Response: Correct your customization and restart the process.

CME1134E Customized queue alarm is in conflict with internal semaphore names. Change the profile tag *Tag*.

Explanation: The customization data is ambiguous.

System Action: The process exits.

User Response: Correct your customization and restart the process.

CME1135E Customized queue alarm name is in conflict with the user ID, or the application name, or both. Check and change the profile tags *Tag1 Tag2*.

Explanation: The customization data is ambiguous.

System Action: The process exits.

User Response: Correct your customization and restart the process.

CME1136E Customized user/process ID is in conflict with internal semaphore names. Change tag *Tag*.

Explanation: The customization data is ambiguous.

System Action: The process exits.

User Response: Correct your customization and restart the process.

CME1137W Unsolicited message received in the queue *Queue*.

Explanation: An unsolicited message has been received in the queue.

System Action: The message is ignored and removed from the queue.

User Response: Check the MQSeries customization on the local and remote systems.

CME1139E Critical error during a call to the operating system: Creation of exit function for *FunctionName* failed (rc = *ReturnCode*). The program will exit immediately.

Explanation: A critical operating system error occurred.

System Action: The process exits.

User Response: Report the error to your IBM representative.

CME1140E Critical error during a call to the PM function: *FunctionName* (PMerror = *ReturnCode*).

Explanation: A critical Presentation Manager error occurred.

System Action: The process exits.

User Response: Report the error to your IBM representative.

CME1141W Message *MsgID* received in the queue *Queue* cannot be correlated.

Explanation: The message *MsgID* could not be found in the reference queue *Queue*.

System Action: The message is ignored and removed from the queue.

User Response: Check the MQSeries customization on the local and remote systems.

CME1142E Intermediate file *File* already exists.

Explanation: The intermediate file *File* already exists in the import directory. The import file can therefore not be processed. This happens if the same import file name is used more than once.

System Action: The program terminates.

User Response: Find out why import files with the same name are generated.

CME1150E Error during a call to the function *FunctionName* : Standard input is not a tty.

Explanation: The Import/Export process tries to read from the starting terminal (stdin) that should be a tty but is not.

System Action: The Import/Export process exits.

User Response: Start the Import/Export process as a foreground process.

CME1155E The profile *FileName* was not found.

Explanation: The profile pointed by *FileName* was not found.

System Action: The process exits.

User Response: Check the path and the file name of the profile.

CME1156E No profile is specified.

Explanation: The program was called without a profile name as parameter.

System Action: The process exits.

User Response: Restart the program with a valid profile name.

CME1160E Not enough disk space for the log file; the process will exit.

Explanation: There is not enough space left on the disk for logging messages.

System Action: The process exits.

User Response: Purge unused files on your disk or define the log file path to another file system.

CME1200E License not found; the process will exit.

Explanation: You do not have enrolled a valid license.

System Action: The process exits.

User Response: Enroll a valid license from the delivered license key diskette.

CME1201E License is expired; the process will exit.

Explanation: Your enrolled license has expired.

System Action: The process exits.

User Response: Get a new valid license.

CME1202E License start date is in the future; the process will exit.

Explanation: You have enrolled a license the start date of which is in the future.

System Action: The process exits.

User Response: Check your system date. Check your license.

CME1203E No licenses available; the process will exit.

Explanation: You are using too many instances of this application. There are no more licenses available.

System Action: The process exits.

User Response: Close other instances of this application. Check stale licenses with your ″Basic License Tool″, shipped with ″License Use Runtime″.

CME1204E License Problem: No such product; the process will exit.

Explanation: You have no license to use this product.

System Action: The process exits.

User Response: Check your enrolled licenses.

CME1205E Could not reach license server; the process will exit.

Explanation: The application could not reach the license server.

System Action: The process exits.

User Response: Check if the service is running.

CME1206E Time difference too big; the process will exit.

Explanation: The difference between your PC clock and the clock of the license server is more than a day.

System Action: The process exits.

User Response: Check your system clock.

CME1207E Not enough licenses available; the process will exit.

Explanation: Your application needs more licenses.

System Action: The process exits.

User Response: Close another instance of your application to release more licenses.

CME1208E User not authorized for license; the process will exit.

Explanation: This user is not authorized to use the license.

System Action: The process exits.

User Response: Log on as a different user.

CME1209E Could not find the vendor of the license; the process will exit.

Explanation: The vendor of the provided license could not be found.

System Action: The process exits.

User Response: Check your enrolled license.

CME1210E Incorrect license client; the process will exit.

Explanation: The application tried to get a license from the license server but used a key that was different from the key used by the installed license.

System Action: The process exits.

User Response: Check your enrolled license.

CME1211E Per-Seat license is not enabled; the process will exit.

Explanation: The per-seat license is installed but not yet available.

System Action: The process exits.

User Response: Contact your system administrator.

CME1212E Incorrect server; the process will exit.

Explanation: The product attempted to get a license from the license server, but used a key that was different from the key used by the installed license.

System Action: The process exits.

User Response: Check your enrolled license.

CME1213E No valid server found; the process will exit.

Explanation: License servers with licenses for a product of the requested vendor servers were not found. Probably, the license servers with information for the specified vendor are not running.

System Action: The process exits.

User Response: Check the license server with licenses for this application.

CME1214E Shared library not found; the process will exit.

Explanation: The static library of License Use Runtime was unable to load a shared library.

System Action: The process exits.

User Response: Check that License Use Runtime is installed correctly on the client.

CME1215I Running with try and buy license.

Explanation: You are using this application with a try and buy license.

System Action: none.

User Response: none.

CME1898I Customization parameter read from *ProcessType* **Profile** *ParamName ParamValue* **=** *"ProfileName"*

Explanation: The value *ParamValue* of the parameter *ParamName* in the profile *ProfileName* of the MIE *ProcessType:ehp3 process is displayed.*

System Action: none.

User Response: none.

CME1900I The MERVA Import Process is going to shut down.

Explanation: One of the import processes will exit.

System Action: The import process will be stopped.

User Response: None.

CME1901W The MERVA Import Process with process ID *ProcessID* **has been invoked in duplicate. The second instance will be stopped immediately.**

Explanation: An import process is invoked twice with the same MERVA user or application ID, possibly with the same process profile.

System Action: The second instance will be stopped.

User Response: Check the process profile of the second import process and compare its contents with those of the first process' profile.

CME1902I The MERVA Import Process is going to start up.

Explanation: The import process will start.

System Action: The import process will be started.

User Response: None.

CME1903W *ProcessName* **will stop.**

Explanation: The process *ProcessName* will stop.

System Action: The process will be stopped.

User Response: None.

CME1904W *ProcessName* **will stop due to** *Reason***.**

Explanation: The process *ProcessName* will stop. The reason for stopping is reported through *Reason*.

System Action: The process will be stopped.

User Response: None.

CME1915I Start loading customization data of the MERVA Import Process from customization profile *ProfileName***.**

Explanation: The process profile will be read from the file *ProfileName*.

System Action: None.

User Response: None.

CME1916I Customization data of the MERVA Import Process *ProcessID* **is loaded successfully.**

Explanation: The process profile of the process with the MERVA application name *ProcessID* has been read.

System Action: None.

User Response: None.

CME2000E The command line arguments are syntactically invalid.

Explanation: The command line arguments are syntactically invalid.

System Action: The command was not processed.

User Response: Call the program with parameter '-h' to to get more information.

CME2001E Command line arguments are semantically invalid.

Explanation: The command line arguments are semantically invalid.

System Action: The command was not processed.

User Response: Call the program with parameter '-h' to to get more information.

CME2002E Cannot register event logging.

Explanation: Could not make the necessary entries in the Windows registry.

System Action: The command was not processed.

User Response: Increase the size of the registry, reboot the system, and reissue the command.

CME2003W Cannot unregister event logging.

Explanation: No entries could be deleted from the Windows registry.

System Action: The command was not processed.

User Response: Reboot the system and reissue the command.

CME2004E Cannot undo register event logging.

Explanation: No entries could be restored in the Windows registry.

System Action: The command was not processed.

User Response: Reboot the system, remove and reinstall the service.

CME2005E Cannot get access to service due to a general error.

Explanation: The service installation and administration tool has failed to get access to the service from the Windows service control manager (SCM).

System Action: The command was not processed.

User Response: Reboot the system. If the error continues, contact your IBM service representative.

CME2006E Cannot get access to the service control manager (SCM).

Explanation: The service installation and administration tool cannot get access to the Windows service control manager (SCM).

System Action: The command was not processed.

User Response: Reboot the system. If the error continues, contact your IBM service representative.

CME2007E Cannot open the service control manager (SCM) database due to a general error.

Explanation: The service installation and administration tool cannot get access to the Windows service control manager (SCM) database.

System Action: The command was not processed.

User Response: Reboot the system. If the error

continues, contact your IBM service representative.

CME2008E Cannot get access to service due to an invalid service name.

Explanation: The service control manager (SCM) has rejected the specified service name.

System Action: The command was not processed.

User Response: Repeat the command with a different service name.

CME2009I Service successfully installed.

Explanation: The service has been successfully installed to the Windows service control manager (SCM) database.

System Action: None.

User Response: None.

CME2010E Cannot create service due to a circular service dependency.

Explanation: The Windows service control manager (SCM) cannot find a valid sequence to start all dependent services.

System Action: The command was not processed.

User Response: Check the specified dependency list and remove the circular dependency.

CME2011E Cannot create service due to an invalid display name.

Explanation: The Windows service control manager (SCM) has rejected the specified service display name.

System Action: The command was not processed.

User Response: Reissue the command with a different service display name.

CME2012E Cannot create service due to an invalid parameter.

Explanation: The Windows service control manager (SCM) has rejected a specified service creation parameter.

System Action: The command was not processed.

User Response: Reissue the command with different parameters.

CME2013E Cannot create service due to an invalid user account.

Explanation: The specified user account does not exist or has not the Windows system right ″Load and unload device drivers″ or ″Log on as a service″.

System Action: The command was not processed.

User Response: Reissue the command using a different user account, or update the corresponding user rights.

CME2014E Cannot create service because a service with the same name already exists.

Explanation: Cannot create a service because a service with the same name already exists.

System Action: The command was not processed.

User Response: Reissue the command with a different service name.

CME2015E Cannot create service due to an invalid service name.

Explanation: The service control manager (SCM) rejected the specified service name.

System Action: The command was not processed.

User Response: Reissue the command with a different service name.

CME2016I Service successfully removed.

Explanation: The service was successfully removed from the Windows service control manager (SCM) database.

System Action: None.

User Response: None.

CME2017E Cannot get access to service because service does not exist.

Explanation: The Windows service control manager (SCM) does not know this service.

System Action: The command was not processed.

User Response: Reissue the command with an existing service name.

CME2018E Cannot remove service because service is still running.

Explanation: The service must be stopped before it can be removed.

System Action: The command was not processed.

User Response: Stop the service and reissue the command.

CME2019W Service already marked for removal.

Explanation: The service was marked for deletion more than once.

System Action: None.

User Response: None.

CME2020I Service has started.

Explanation: A start command for a service was sent to the service control manager (SCM).

System Action: The service process will start.

User Response: To check whether the service is running, reissue the service installation and administration program with the '-qss' argument.

CME2021E Cannot start service because the service executable file cannot be found.

Explanation: The service executable was not found in the directories specified by the PATH environment variable.

System Action: The command was not processed.

User Response: Update the service with a fully-qualified service path, or add the service directory to the PATH environment variable.

CME2022E Cannot start service because service is marked for removal.

Explanation: The Windows service control manager (SCM) cannot start a service that has been marked for removal.

System Action: The command was not processed.

User Response: Install and then restart the service again.

CME2023E Cannot start service because service is already running.

Explanation: A service cannot be started twice.

System Action: The command was not processed.

User Response: None.

CME2024E Cannot start service due to invalid service dependencies.

Explanation: The service control manager (SCM) failed to start the dependent services.

System Action: The command was not processed.

User Response: Try to start all dependent services manually and reissue the command.

CME2025E Cannot start service because service has been disabled.

Explanation: The Windows service control manager (SCM) cannot start a service that has been disabled.

System Action: The command was not processed.

User Response: Enable the service with the Services applet and reissue the command.

CME2026E Cannot start service because service cannot be logged on.

Explanation: The Windows service control manager (SCM) cannot log on the service with the specified user account.

System Action: The command was not processed.

User Response: Check whether the specified user account has the right "Log on as service". If not, add this right to the account.

CME2027E Cannot start service due to a thread creation error.

Explanation: A new thread could not be created to run the service.

System Action: The command was not processed.

User Response: Reboot the system and reissue the command.

CME2028E Cannot start service due to a service initialization error.

Explanation: The service did not signal a successful start to the Windows service control manager (SCM) within the necessary time interval.

System Action: The command was not processed.

User Response: Check whether the service is installed with the correct parameters. If not, update the service installation. If so, reboot the system and reissue the command.

CME2029I Service has terminated.

Explanation: The Windows service control manager (SCM) sent a termination request to the service.

System Action: None.

User Response: To check whether the service has terminated, reissue the service installation and administration program with the '-qss' argument.

CME2030E Service cannot be stopped because other running services are dependent on it.

Explanation: The service cannot be stopped because other running services are dependent on it.

System Action: The command was not processed.

User Response: Terminate all dependent services, then reissue the command.

CME2031E Requested control code cannot be sent to the service.

Explanation: The requested control code cannot be sent to the service because the service has stopped or is pending.

System Action: The command was not processed.

User Response: Wait until the service has left its pending state. When service is running, reissue the command.

CME2032E Cannot stop service because service is not active.

Explanation: The service has not been started.

System Action: The command was not processed.

User Response: None.

CME2033E Cannot stop service because the system is shutting down.

Explanation: The Windows service control manager (SCM) cannot sent a termination request to the service because the system is shutting down.

System Action: The command was not processed.

User Response: None.

CME2034E Cannot stop service due to a general error.

Explanation: The Windows service control manager (SCM) could not sent a termination request to the service.

System Action: The command was not processed.

User Response: To terminate the service, reboot the system.

CME2035I Service was successfully updated.

Explanation: The service was successfully updated.

System Action: None.

User Response: None.

CME2036E Cannot update service due to a circular dependency.

Explanation: The service control manager (SCM) cannot find a valid sequence to start all dependent services.

System Action: The command was not processed.

User Response: Check the specified dependency list and remove the circular dependency.

CME2037E Cannot update service due to an invalid display name.

Explanation: The Windows service control manager (SCM) has rejected the specified service display name.

System Action: The command was not processed.

User Response: Reissue the command with a different service display name.

CME2038E Cannot update service due to an invalid parameter.

Explanation: The service control manager (SCM) has rejected a specified service creation parameter.

System Action: The command was not processed.

User Response: Reissue the command with different parameters.

CME2039E Cannot update service due to an invalid user account.

Explanation: The specified user account does not exist or has not the Windows system right ″Load and unload device drivers″ or ″Log on as a service″.

System Action: The command was not processed.

User Response: Reissue the command using a different user account, or update the corresponding user rights.

CME2040E Cannot update service because service is marked for removal.

Explanation: The service control manager (SCM) cannot update a service that is marked for removal.

System Action: The command was not processed.

User Response: Reinstall and then start the service.

CME2041W Message catalog (DLL) not registered.

Explanation: No message catalog was registered to the Windows registry for service event logging.

System Action: None.

User Response: If a message catalog should be registered, update the service using the ’-sm’ Parameter.

CME2042E User account *User_account* does not exist.

Explanation: The specified user account does not exist.

System Action: The command was not processed.

User Response: Reissue the command with a valid user account.

CME2043E Due to an access denial, cannot check if user exists.

Explanation: Due to an access denial, cannot check if user exists.

System Action: The command was not processed.

User Response: Check user account and reboot the system.

CME2044E Due to an invalid computer name, cannot check if user exists.

Explanation: The service installation program tried to check whether your user ID exists on your system.

System Action: The command was not processed.

User Response: Contact your IBM service representative.

CME2045E Due to a general error, cannot check if user exists.

Explanation: Due to a general error, cannot check if user exists.

System Action: The command was not processed.

User Response: Reboot the system and reissue the command.

CME2070E Cannot start service *serviceName* **due to invalid data.**

Explanation: Cannot start the specified service due to invalid data.

System Action: The service process will terminate.

User Response: Contact your IBM representative.

CME2071E Cannot start service *serviceName* **because service is already running.**

Explanation: Cannot start the specified service because the service is already running.

System Action: None.

User Response: None.

CME2072E Cannot start service *serviceName* **due to invalid parameters.**

Explanation: The service control manager (SCM) has called the service with invalid parameters.

System Action: The service process will terminate.

User Response: Check your installed service instance using the '-qs' parameter. Update your service instance with valid service parameters using the '-sp' parameter.

CME2073I Service *serviceName* **successfully started.**

Explanation: The service was successfully started.

System Action: The service process is running.

User Response: None.

CME2074E Cannot start service *serviceName***.**

Explanation: The specified service cannot be started.

System Action: The service process will terminate.

User Response: See the MIE log file for more information. If the log file contains no error messages, check whether the log file was locked by another process.

CME2075E A severe error occurred when running the service *serviceName***.**

Explanation: A severe error occurred when running the specified service.

System Action: The service process will terminate.

User Response: See the MIE log file for more information.

CME2076E A service handle for service *serviceName* **could not be opened or created.**

Explanation: A service handle could not be opened or created for the specified service.

System Action: The service process will terminate.

User Response: Reboot the system and restart the service.

CME2077E The status of the service *serviceName* **could not be sent to the Windows service control manager (SCM).**

Explanation: The status of the specified service could not be sent to the Windows service control manager (SCM).

System Action: The service process will terminate.

User Response: Reboot the system and restart the service.

CME2078E No password is specified in the profile for service *serviceName***.**

Explanation: The password string CMEMIE_EX_PASS in your export profile or CMEMIE_IM_PASS in your import profile, respectively, is missing or empty.

System Action: The service process will terminate.

User Response: Insert a password in the profile and restart the service.

CME2079E Service *serviceName* cannot write message into log file.

Explanation: The service has failed to write a log message into the log file. If no log file is defined by the environment variable CMEMIELOG the service tries to write to the Windows system32 directory.

System Action: The service process will terminate.

User Response: Check, if the service has write access to the log file and its directory. Check, if the space on the file system is full.

CME2100E The specified profile has an invalid format.

Explanation: The encryption utility has recognized that the content of the profile is not in the expected format.

System Action: The encryption utility will terminate.

User Response: Check whether the name of the specified profile is correct. If so, lookup the user manual for more information about the format of the profile.

CME2101E Command line arguments are syntactically invalid.

Explanation: The command line arguments are syntactically invalid.

System Action: The encryption utility will terminate.

User Response: Call the program with parameter '-h' to to get more information.

CME2102E Unable to open specified profile.

Explanation: Unable to open specified profile. The profile was not found or was read-protected or write-protected.

System Action: The encryption utility will terminate.

User Response: Check if path name and profile name are correct or remove read-protection or write-protection.

CME2103E No profile filename specified.

Explanation: The user did not specify a profile filename.

System Action: The encryption utility will terminate.

User Response: Call the program with parameter '-h' to get more information.

CME2104E You entered different values.

Explanation: The user entered two different keyword values.

System Action: The user is asked again to enter the keyword value (e.g. the password) twice.

User Response: Re-enter the value twice.

CME2105E You did not enter a user ID.

Explanation: The user did not enter a user ID.

System Action: The user is asked again to enter a valid user ID.

User Response: Enter a valid user ID.

CME2106E You did not enter a password.

Explanation: The user did not enter a password.

System Action: The user is asked again to enter a valid password.

User Response: Enter a valid password.

CME2107E You did not enter a value.

Explanation: User did not enter a value.

System Action: The user is asked again to enter a valid keyword value.

User Response: Enter a valid keyword value.

CME2108E You entered a user ID that is longer than *idlength* characters.

Explanation: The user entered a user ID that is longer than 8 characters.

System Action: The user is asked to enter a new user ID.

User Response: Enter a new user ID, that does not exceed 8 characters.

CME2109E You entered a password that is longer than *passwdlength* characters.

Explanation: The user entered a password that is longer than 8 characters.

System Action: The user is asked to enter a new password.

User Response: Enter a new password, that does not exceed 8 characters.

CME2110I Enter your user ID:

Explanation: The user is requested to enter a user ID.

System Action: The system waits for user input.

User Response: Enter a user ID, that does not exceed the maximum length.

CME2111I Enter your password:

Explanation: The user is requested to enter a password.

System Action: The system waits for user input.

User Response: Enter a password, that does not exceed the maximum length.

CME2112I Keyword does not exist or contains no data. Enter the value to be encrypted:

Explanation: The user is requested to enter the value of the keyword.

System Action: The system waits for user input.

User Response: Enter the value of the keyword, that is to be encrypted.

CME2113I Re-enter your password:

Explanation: The user is requested to re-enter the password.

System Action: The system waits for user input.

User Response: Re-enter the password.

CME2114I Re-enter your keyword value:

Explanation: The user is requested to re-enter the value of the keyword.

System Action: The system waits for user input.

User Response: Re-enter the value of the keyword.

CME2115I Password successfully encrypted.

Explanation: The password was successfully encrypted, and the parameter CMEMIE_IM_PENC (for an import profile) or CMEMIE_EX_PENC (for an export profile) was added to the profile to indicate this.

System Action: None.

User Response: None.

CME2117W Password already encrypted.

Explanation: The user tried to encrypt a password in a profile, but that password was already encrypted.

System Action: The encrypted password is left as it was.

User Response: None.

Appendix H. 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 or other countries or both:

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

- MVS/ESA
- MVS/XA
- OS/2
- OS/390
- RACF
- 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 partener 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.

ASP. (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 DSLLUNIT and DSLLUEND macroinstruction.

UNIX System Services (USS). A component of OS/390, formerly called OpenEdition (OE), that creates a UNIX environment that conforms to the XPG4 UNIX 1995 specifications, and provides two open 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 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

L

M

N

O

P

Q

R

S

T

U

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

MERVA ESA Components
MERVA Automatic Message Import/Export Facility
User's Guide
Version 4 Release 1

Publication No. SH12-6389-02

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

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

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

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SH12-6389-02

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Deutschland Entwicklung GmbH
Information Development
Dept. 0446
Schoenaicher Strasse 220
71032 Boeblingen
Germany

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Program Number: 5648-B30

SH12-6389-02

Spine information:

MERVA ESA Components

MERVA Automatic Message Import/Export Facility
User's Guide

Version 4
Release 1