MERVA Family

MERVA Connection/ESA

SH12-6187-00

MERVA Family

MERVA Connection/ESA

SH12-6187-00

Note!

Before using this information and the product it supports, be sure to read the general information under “Appendix E. Notices” on page 95.

First Edition, November 1997

This edition applies to

Version 3 Release 3 of IBM MERVA for OS/2 LAN (5622-122)

Version 3 Release 3 of IBM MERVA for OS/2 Standalone (5622-127)

Version 1 Release 2 of IBM MERVA for AIX (5765-449)

and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About This Book

This book is intended for application programmers who want to connect an IBM S/390 application to the Message Entry and Routing with Interfaces to Various Applications for OS/2 Version 3 (abbreviated to MERVA OS/2 V3 in this book) or the Message Entry and Routing with Interfaces to Various Applications for AIX (abbreviated to MERVA AIX in this book). It is also for users who want to install and customize MERVA Connection/ESA.

It is assumed that you have prior knowledge of and experience with:

- S/390
- MVS/ESA
- CICS/ESA
- Operating System/2 (OS/2)
- Advanced Interactive Executive (AIX)
- RISC/6000 (RS/6000)
- Systems Network Architecture (SNA)
- Application Programming Interface (API) of MERVA OS/2 V3
- Application Programming Interface of MERVA AIX.

Chapter 1. Introduction to MERVA Connection/ESA

This chapter introduces the MERVA Connection/ESA and briefly describes the facilities supported by MERVA Connection/ESA.

The Objectives of MERVA Connection/ESA

There is a wide range of banking applications available for the S/390 platform.

While many of these applications create and process SWIFT messages, they do not provide a connection to public networks. MERVA OS/2 V3 and MERVA AIX provide connections to the SWIFT network (with SWIFT Link), to the public telex network (only MERVA OS/2 V3 with TelexPlus), and to other MERVA OS/2 V3, MERVA AIX, or MERVA ESA systems (with MERVA Link).

To use S/390 applications as banking applications, messages created on the S/390 must be transferred to MERVA OS/2 V3 or MERVA AIX. Messages received from one of these networks must be transferred from MERVA OS/2 V3 or MERVA AIX to the S/390.

While this can be achieved by saving messages to files and transferring the files, this solution requires operator intervention and can cause message integrity problems. It may also not be transparent to the application. Therefore, the best method is to implement a direct connection from the application on the S/390 to MERVA OS/2 V3 or MERVA AIX, as if MERVA OS/2 V3 or MERVA AIX was a component of the application.

MERVA Connection/ESA is a tool that makes it easy for you to implement such a solution. MERVA Connection/ESA is not a ready-to-use SWIFT interface on the S/390. It does not have a user interface.

It is an interface for application programs on the S/390. With a minimum of effort you can create an application (remote API) on the S/390 to send messages to MERVA OS/2 V3 or MERVA AIX and receive messages from there using MERVA Connection/ESA.

Functions Provided by MERVA Connection/ESA

MERVA Connection/ESA provides the complete functionality of the MERVA OS/2 V3 and MERVA AIX API on the S/390. Additional calls are available for establishing the connection and making use of MERVA OS/2 V3 or MERVA AIX alarms. MERVA Connection/ESA provides a real-time interface to MERVA OS/2 V3 or MERVA AIX.

Environment

MERVA Connection/ESA can be run under CICS/ESA in the MVS/ESA environment.

Language Support

Easy portability of MERVA OS/2 V3 API programs between OS/2 and CICS/ESA and MERVA AIX API programs between AIX and CICS/ESA is provided by the C Language interface. Consider, however, that under CICS/ESA there is no support by the C/370 I/O library for disk files and other devices.

Security

Security aspects are dealt with by a very flexible user exit interface (see “Chapter 8. Security” on page 55).

Message Integrity

A resynchronization mechanism ensures that the remote API program can provide the same level of message integrity as a local API program.

Principles

Figure 1 illustrates the main programming concepts of MERVA Connection/ESA.

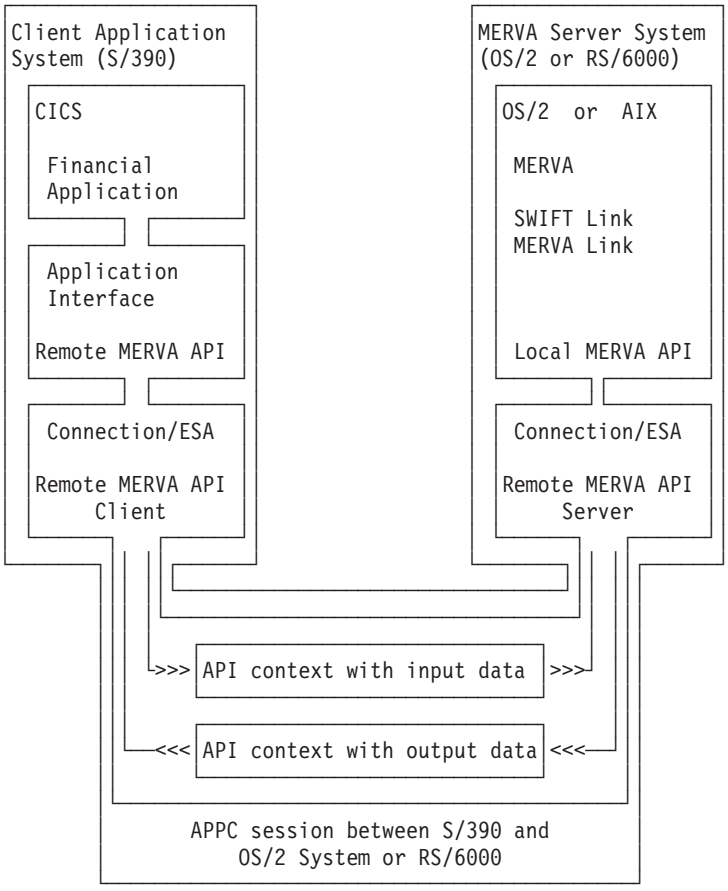

Figure 1. Concept of MERVA Connection/ESA

The Remote MERVA API Client provides the calling interface for the application on the S/390. It forwards the API call with the input parameters to the Remote MERVA API Server on the OS/2 or AIX system. The Remote MERVA API Server calls the MERVA OS/2 V3 or MERVA AIX API function and passes the received parameters to it. The output data and return code are transferred back to the Remote MERVA API Client. There, control is passed back to the calling program as if the API function had been executed locally.

Chapter 2. Planning and Preparing for Installing MERVA Connection/ESA

This chapter describes the machine and software requirements that are prerequisites for installing MERVA Connection/ESA.

Machine Requirements

MERVA Connection/ESA can communicate with a MERVA OS/2 V3 stand-alone installation, a workstation within a MERVA OS/2 V3 LAN installation or an instance of a MERVA AIX installation. For OS/2 system hardware requirements, see the *MERVA OS/2 V3 Installation and Customization Guide*. For AIX hardware requirements, see the *MERVA AIX Installation and Customization Guide*. Depending on the type of connection used (see below), a Token Ring card or a multi-protocol adapter (MPA) card is required.

MERVA Connection/ESA requires an APPC LU 6.2 session between the OS/2 system and the S/390 or between the RS/6000 and the S/390. This can be achieved by Token Ring, SDLC, and other types of connection for which the Communications Server, the SNA Server and VTAM provide support for an LU 6.2 session. For a list of the alternatives available and the hardware required for each, refer to the appropriate books listed in the “Bibliography” on page 101.

Program Requirements

The following describes the software that is required on the OS/2 system, the RS/6000 and the S/390 machines.

Programs Required on the MERVA OS/2 V3 Side

- Communications Server for OS/2 V4.0, or a subsequent release

  or

  Personal Communications V4.1, or a subsequent release.

Programs Required on the MERVA AIX Side

- SNA Server for AIX V3.1, or a subsequent release

  or

  Communications Server for AIX V4.1, or a subsequent release.

Programs Required on the S/390 Side

The following software must be installed on the S/390:

- MVS/SP Version 3 Release 1.3 (MVS/ESA), or a subsequent release
- CICS/ESA Version 3 Release 2.1, or a subsequent release
- ACF/VTAM Version 3 Release 4 for MVS/ESA, or a subsequent release
- ACF/NCP Version 4 Release 3.1, or a subsequent release
- C/370 Library Version 2 Release 2, or a subsequent release

  or

LE/370 Version 1 Release 3, or a subsequent release, together with CICS/ESA Version 3 Release 3, or a subsequent release

- AD/Cycle C/370 Compiler Version 1 Release 2, or a subsequent release.

Network Requirements

The network connection between the S/390 and the OS/2 system or the RS/6000 must support an LU 6.2 session. For establishing an LU 6.2 session between an OS/2 system and an S/390, the appropriate profiles in the Communications Server and the definitions in VTAM and CICS/ESA must be customized. For establishing an LU 6.2 session between an RS/6000 and an S/390, the appropriate profiles in the SNA Serverand the definitions in VTAM and CICS/ESA must be customized.

Chapter 3. Installing MERVA Connection/ESA

This chapter describes how to install MERVA Connection/ESA on the OS/2 system, the RS/6000, and the S/390.

Installing MERVA Connection/ESA on the OS/2 System

The installation of MERVA Connection/ESA on the OS/2 system must be done if you want to use MERVA OS/2 V3.

The following describes what you must do to install the Remote MERVA API Server program and sample Communications Server configuration files on the OS/2 system.

Installing the Remote MERVA API Server Program

The Remote MERVA API Server program is installed by MERVA OS/2 V3.

To replace the sample security user exits, you must copy a source file from the MERVA OS/2 V3 samples diskette. You can either copy it to the *x*:\MERVA2\BASE directory, or to a directory of your choice. Use the following command:

copy A:\CONNECT\ENM4SSEC.C ***x*:*****directory***

where *x* is the drive where MERVA OS/2 V3 is installed and *directory* is the name of the directory of your choice.

Installing the Sample Communications Server Configuration Files

The MERVA OS/2 V3 samples diskette contains a sample set of Communications Server configuration files. If you want to use these, unzip them to the directory where Communications Server is installed:

PKUNZIP2 A:\CONNECT\ES9000\ES9000.ZIP ***x*:\CMLIB**

where *x* is the drive Communications Server is installed on (CMLIB directory).

Change the name of the default configuration file in Communications Server to **ES9000** by double-clicking on the icon Replace Default Configuration of the Communications Server folder.

Installing MERVA Connection/ESA on the RS/6000

The installation of MERVA Connection/ESA on the RS/6000 must be done if you want to use MERVA AIX.

The Remote MERVA API Server program is already installed by MERVA AIX.

Installing the SNA Server Profiles

The connection between the remote application side and the MERVA AIX side must be an LU 6.2 session. To install the required SNA Server profiles, do the following:

- Create the required SNA Server profiles manually using the System Management Interface Tool (SMIT). Samples for the network definitions are shown in “Appendix B. Sample Network Definitions for the RS/6000” on page 79.

Installing MERVA Connection/ESA on the S/390

A tape contains the libraries that make up the MERVA Connection/ESA system. The tape is a standard label tape with the label **0HG330**. It is installed using SMP/E.

It contains the following files:

SMPMCS	MCS Statements
IBM.H0HG330.F1	Installation Library installed as &PRFX.SENMINS0
IBM.H0HG330.F2	Source Library installed as &PRFX.SENMSRC0
IBM.H0HG330.F3	Macro Library installed as &PRFX.SENMMAC0
IBM.H0HG330.F4	Object Library installed as &PRFX.SENMOBJ0

During the installation process you replace &PRFX by the high level qualifier(s) of your choice for the library name. At the end of the installation, the libraries contain the following members:

&PRFX.SENMINS0:	**Installation Library**
ENMAC05	Prelink and link-edit procedure for MERVA Connection/ESA API programs.
ENMCCICS	Compile and link-edit procedure for MERVA Connection/ESA API programs containing EXEC CICS statements.
ENMCCOMP	Compile and link-edit procedure for MERVA Connection/ESA API programs containing C/370 code only.
ENMFILES	Allocation of the MERVA Connection/ESA sequential data sets and loading of the connection profile data set.
ENMMICTL	Allocation of the MERVA Connection/ESA message integrity VSAM file.
&PRFX.SENMSRC0:	**Source Library**
ENMABEND	Abnormal termination exit for the MERVA Connection/ESA installation verification program ENMVERIF.
ENMSAMP	Sample API application for loading telex messages to a MERVA OS/2 V3 queue.
ENMVERIF	MERVA Connection/ESA installation verification program.
ENM4SNIL	Skeleton for MERVA Connection/ESA security exits.
ENM4SSEC	Sample for MERVA Connection/ESA security exits.
&PRFX.SENMMAC0:	**Macro Library**
ENMAPPL	VTAM definition of the CICS Application ID.

ENMCSD	CICS/ESA resource definitions to be used as input for the utility DFHCSDUP.
ENMDCT	CICS/ESA Destination Control Table (DCT) definitions.
ENMLOGMO	VTAM Logmode entries for an APPC connection.
ENMPULU	VTAM PU/LU definition of an OS/2 system in a Token Ring.
ENM4RAPI	Include-file for a MERVA Connection/ESA API program containing required definitions.
ENM4SXIT	Include-file for MERVA Connection/ESA security exits containing required definitions.
&PRFX.SENMOBJ0:	**Object Library**
ENMABEN	Abnormal termination exit for the MERVA Connection/ESA installation verification program.
ENMRAPI	MERVA Connection/ESA API.
ENMRPRF	MERVA Connection/ESA profile module.
ENMRUTL	MERVA Connection/ESA utility programs.
ENMSAM	Sample API application for loading telex messages to a MERVA queue.
ENMSNIL	Skeleton for MERVA Connection/ESA security exits.
ENMSSEC	Sample for MERVA Connection/ESA security exits.
ENMVERI	MERVA Connection/ESA installation verification program.

The load modules for the MERVA Connection/ESA sample API programs will be prelink-edited and link-edited during the installation process (refer to page 17). The resulting load modules are contained in a library with the low level qualifier SENMLOD0:

&PRFX.SENMLOD0:	**Load Library**
ENMABEND	Abnormal termination exit for the MERVA Connection/ESA installation verification program ENMVERIF.
ENMSAMP	Sample API application for loading telex messages to a MERVA queue.
ENMVERIF	MERVA Connection/ESA installation verification program.

The function modification identifier (FMID) and the component identifier (COMPID) are:

FMID	H0HG330
COMPID	562212208
Component Name	MERVA Connection/ESA
REL	330

Installation Steps

The installation steps are summarized as follows:

Step 0 Unload the installation JCL from the MERVA Connection/ESA distribution tape.

Step 1 Allocate distribution and target libraries.

Step 2 Allocate SMP/E data sets and SMP/E CSI.

Step 3 Initialize the SMP/E zones.

Step 4 Carry out the RECEIVE and APPLY processing for MERVA Connection/ESA.

Step 5 Prelink-edit and link-edit the MERVA Connection/ESA sample API programs.

Step 6 Carry out the ACCEPT processing for MERVA Connection/ESA.

Beginning with step 0, the provided sample jobs are sequentially numbered in the sequence in which you should run them after you have modified them as described in the following.

The sample jobs must be modified according to your installation requirements. The statements to be modified are marked by MODIFY in columns 65-70 on the installation library.

The MODIFY instructions given in the control statements for VSAM Access Method Services jobs and SMP/E jobs would render the affected jobs syntactically invalid and therefore must be deleted before running.

Step 0 - Unload the Installation JCL

To unload the installation JCL from the MERVA Connection/ESA distribution tape you need to generate an unload job.

There are two methods to generate the job:

- If a 3270 emulator session to TSO is available on your OS/2 system, you can copy job UNLOAD.JOB from the MERVA OS/2 V3 samples diskette to an existing partitioned data set which has a logical record length (LRECL) of 80. Logon to TSO, then enter the following command on the OS/2 System:

  send A:\CONNECT\ES9000\UNLOAD.JOB *id*:'*dsn*(UNLOAD)' ASCII CRLF

  where *id* is the 3270 emulator session used for the file transfer, and *dsn* is the fully qualified name of the partitioned data set.
- If you cannot transfer job UNLOAD.JOB, type in the job shown in Figure 2 on page 9. Change all values indicated by MODIFY as required by your local environment.

```
//JOBNAME  JOB (ACCT),NAME,MSGCLASS=L,CLASS=Z                   MODIFY
//*****************************************************************
//* THIS JOB ALLOCATES THE CONNECTION/ESA INSTALLATION JCL LIBRARY
//* AND UNLOADS INSTALLATION JCL FROM TAPE INTO THIS LIBRARY.
//* ---------------------------------------------------------
//* PLEASE REMOVE COMMENTS FROM CONTROL CARDS BEFORE JOB START
//* ---------------------------------------------------------
//*****************************************************************
//UNLOAD   PROC JCLLIB='CONN.JCLLIB',TAPUNIT=TAPE,UNIT=SYSDA    MODIFY
//CPYJCL   EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//IN1      DD DSN=IBM.H0HG330.F1,
//            UNIT=&TAPUNIT.,VOL=SER=0HG330,
//            LABEL=(2,SL),
//            DISP=(SHR,KEEP)
//OUT1     DD DSN=&JCLLIB.,
//            UNIT=&UNIT.,
//            DISP=(NEW,CATLG,DELETE),
//            SPACE=(6160,(16,8,2)),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//      PEND
//      EXEC UNLOAD
//CPYJCL.SYSIN DD *
  COPY INDD=IN1,OUTDD=OUT1
/*
//
```

Figure 2. Job to Unload the MERVA Connection/ESA Installation JCL

Submit the job.

When this job completes successfully with a condition code of 0, the data set you specified in parameter JCLLIB contains the SMP/E JCL.

Run the jobs that are described in more detail in the following sections after modifying the values indicated by MODIFY in columns 65-70 on the installation library as required by your local environment. If an installation job completes abnormally, you must find out why it failed from the error messages produced in the job log list. Correct the error and rerun the failing job. Do not attempt to run the next job until the previous job has run successfully.

Step 1 - ENMAC01 - Allocate Distribution and Target Libraries

This sample job (member ENMAC01) allocates the MERVA Connection/ESA Distribution and Target Libraries.

```
//ENMAC01  JOB (ACCT),NAME,MSGCLASS=L,CLASS=Z                   MODIFY
//*************************************************************
//* THIS JOB ALLOCATES THE DISTRIBUTION AND TARGET LIBRARIES
//*
//*     MODIFY THE PARAMETERS PRFX, UNIT, AND THE BLKSIZE
//*       PARAMETER IN DCBFB3, DCBFB6, AND DCBU UP TO YOUR
//*       REQUIREMENTS
//* -----------------------------------------------------------
//* PLEASE REMOVE COMMENTS FROM CONTROL CARDS BEFORE JOB START
//* -----------------------------------------------------------
//*************************************************************
//ALLOC  PROC PRFX='CONN',                                      MODIFY
//             UNIT=SYSDA,                                      MODIFY
//             DCBFB3='(RECFM=FB,LRECL=80,BLKSIZE=3200)',       MODIFY
//             DCBFB6='(RECFM=FB,LRECL=80,BLKSIZE=6160)',       MODIFY
//             DCBU='(RECFM=U,BLKSIZE=6144)',                   MODIFY
//             DISP='(NEW,CATLG,DELETE)'
//IEFBR14 EXEC PGM=IEFBR14
//DD01    DD DSN=&PRFX..AENMINS0,DISP=&DISP.,
//           SPACE=(6160,(8,1,1)),
//           UNIT=&UNIT.,DCB=&DCBFB6.
//DD02    DD DSN=&PRFX..AENMSRC0,DISP=&DISP.,
//           SPACE=(6160,(16,2,1)),
//           UNIT=&UNIT.,DCB=&DCBFB6.
//DD03    DD DSN=&PRFX..AENMMAC0,DISP=&DISP.,
//           SPACE=(6160,(16,2,1)),
//           UNIT=&UNIT.,DCB=&DCBFB6.
//DD04    DD DSN=&PRFX..AENMOBJ0,DISP=&DISP.,
//           SPACE=(3200,(100,10,1)),
//           UNIT=&UNIT.,DCB=&DCBFB3.
//DD05    DD DSN=&PRFX..SENMINS0,DISP=&DISP.,
//           SPACE=(6160,(8,1,1)),
//           UNIT=&UNIT.,DCB=&DCBFB6.
//DD06    DD DSN=&PRFX..SENMSRC0,DISP=&DISP.,
//           SPACE=(6160,(16,2,1)),
//           UNIT=&UNIT.,DCB=&DCBFB6.
//DD07    DD DSN=&PRFX..SENMMAC0,DISP=&DISP.,
//           SPACE=(6160,(16,2,1)),
//           UNIT=&UNIT.,DCB=&DCBFB6.
//DD08    DD DSN=&PRFX..SENMOBJ0,DISP=&DISP.,
//           SPACE=(3200,(100,10,1)),
//           UNIT=&UNIT.,DCB=&DCBFB3.
//DD09    DD DSN=&PRFX..SENMLOD0,DISP=&DISP.,
//           SPACE=(6144,(60,6,1)),
//           UNIT=&UNIT.,DCB=&DCBU.
//      PEND
//      EXEC ALLOC
```

Figure 3. Allocate Distribution and Target Libraries

Step 2 - ENMAC02 - Allocate SMP/E Data Sets and SMP/E CSI

This sample job (member ENMAC02) is used to allocate and initialize the SMP/E data sets and the SMP/E CSI.

ENMAC02 consists of three steps:

ALLOC allocates all needed SMP/E data sets

DEFINE allocates the SMP/E CSI

INIT initializes the SMP/E CSI

```
//ENMAC02  JOB (ACCT),NAME,MSGCLASS=L,CLASS=Z                   MODIFY
//*************************************************************
//* THIS JOB ALLOCATES THE CONNECTION/ESA SMP/E DATA SETS,
//* THE SMP/E CSI DATA SET AND INITIALIZES THE SMP/E CSI.
//*     MODIFY THE PARAMETERS PRFX, UNIT, AND THE BLKSIZE
//*       PARAMETER IN DCBFB UP TO YOUR REQUIREMENTS
//*
//*       ENSURE THAT THOSE PARAMETERS THAT HAVE DEFAULT
//*       VALUES ARE REASONABLE FOR YOUR PURPOSES.
//* -----------------------------------------------------------
//* PLEASE REMOVE COMMENTS FROM CONTROL CARDS BEFORE JOB START
//* -----------------------------------------------------------
//*************************************************************
//ALLOCE   PROC PRFX='CONN',                                    MODIFY
//             UNIT=SYSDA,                                      MODIFY
//             DCBFB='(RECFM=FB,LRECL=80,BLKSIZE=6160)',        MODIFY
//             DISP='(NEW,CATLG,DELETE)'
//         EXEC PGM=IEFBR14
//SMPSCDS DD DSN=&PRFX..SMPSCDS,DISP=&DISP.,
//           SPACE=(6160,(100,10,80)),
//           UNIT=&UNIT.,DCB=&DCBFB.
//SMPMTS  DD DSN=&PRFX..SMPMTS,DISP=&DISP.,
//           SPACE=(6160,(10,1,10)),
//           UNIT=&UNIT.,DCB=&DCBFB.
//SMPPTS  DD DSN=&PRFX..SMPPTS,DISP=&DISP.,
//           SPACE=(6160,(350,35,80)),
//           UNIT=&UNIT.,DCB=&DCBFB.
//SMPSTS  DD DSN=&PRFX..SMPSTS,DISP=&DISP.,
//           SPACE=(6160,(10,1,10)),
//           UNIT=&UNIT.,DCB=&DCBFB.
//SMPLOG  DD DSN=&PRFX..SMPLOG,DISP=&DISP.,
//           SPACE=(3200,(250,250)),
//           UNIT=&UNIT.,DCB=(BLKSIZE=3200,RECFM=VB)
//         PEND
//ALLOC    EXEC PROC=ALLOCE
//*
```

Figure 4. Allocate SMP/E Data Sets and SMP/E CSI (Part 1 of 2)

```
//************************************************************
//* THIS STEP ALLOCATES THE SMP/E CSI DATA SET
//* MODIFY THE PARAMETERS
//*        CLUSTER NAME          CONN.SMPCSI.CSI
//*          DATA  NAME          CONN.SMPCSI.CSI.DATA
//*          INDEX NAME          CONN.SMPCSI.CSI.INDEX
//*          UNIT / VOLUME       3390 / CSIVOL
//************************************************************
//DEFINE   EXEC PGM=IDCAMS
//CSIVOL   DD UNIT=3390,VOL=SER=CSIVOL,DISP=SHR                MODIFY
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
   DEFINE CLUSTER(NAME(CONN.SMPCSI.CSI) -                     MODIFY
                  FREESPACE(10 5) -
                  KEYS(24 0) -                                MODIFY
                  RECORDSIZE(24 143) -                        MODIFY
                  SHAREOPTIONS(2) -
                  UNIQUE -
                  VOLUMES(CSIVOL)) -                          MODIFY
          DATA(NAME(CONN.SMPCSI.CSI.DATA) -                   MODIFY
                  CISZ(4096) -
                  CYLINDERS(5 2)) -
          INDEX(NAME(CONN.SMPCSI.CSI.INDEX) -                 MODIFY
                  CYLINDERS(1 1) -
                  IMBED)
/*
//************************************************************
//* THIS STEP INITIALIZES THE SMP/E CSI DATA SET
//************************************************************
//INIT     EXEC PGM=IDCAMS
//ZPOOL    DD   DSN=SYS1.MACLIB(GIMZPOOL),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
   REPRO INFILE(ZPOOL) OUTDATASET(CONN.SMPCSI.CSI)            MODIFY
/*
//
```

Figure 4. Allocate SMP/E Data Sets and SMP/E CSI (Part 2 of 2)

Step 3 - ENMAC03 - Initialize the SMP/E Zones

Before unloading the tape you have to initialize the SMP/E Global Zone, Target Zone, and DLIB Zone as shown in the sample job (member ENMAC03).

The DSSPACE sub-entry in the GLOBAL zone defines the primary, secondary, and directory block allocation required for receiving SYSMODs packaged using RELFILES. DSSPACE specifies the amount of space SMP/E will attempt to allocate for *each* relative file on the RELFILE tape and therefore must be large enough to contain the largest unloaded RELFILE data set.

```
//ENMAC03  JOB (ACCT),NAME,MSGCLASS=L,CLASS=Z                    MODIFY
//*************************************************************
//* THIS JOB ALLOCATES THE SMP/E GLOBAL ZONE
//* DLIBZONE AND TARGETZONE ARE DEFINED AND INITIALIZED
//* -----------------------------------------------------------
//* PLEASE REMOVE COMMENTS FROM CONTROL CARDS BEFORE JOB START
//* -----------------------------------------------------------
//*************************************************************
//UPD EXEC PGM=GIMSMP,REGION=4M,
//            PARM='CSI=CONN.SMPCSI.CSI'                         MODIFY
//SMPLOG  DD DSN=CONN.SMPLOG,DISP=MOD                            MODIFY
//SMPPTS  DD DSN=CONN.SMPPTS,DISP=SHR                            MODIFY
//SMPCNTL DD *
  SET BDY(GLOBAL).
  UCLIN .
         ADD GLOBALZONE
             OPTIONS(ENMOPTN)
             SREL(Z038)
             ZONEINDEX((ENMTGT,CONN.SMPCSI.CSI,TARGET),         MODIFY
                       (ENMDLB,CONN.SMPCSI.CSI,DLIB)) .         MODIFY
         ADD
             OPTIONS(ENMOPTN)
             DSPREFIX(CONN)                                     MODIFY
             DSSPACE(100,10,1)
             PEMAX(9999)
             RETRYDDN(ALL) .
  ENDUCL .
  SET BDY(ENMTGT) .                                             MODIFY
  UCLIN .
         ADD TARGETZONE(ENMTGT)                                 MODIFY
             RELATED(ENMDLB)                                    MODIFY
             OPTIONS(ENMOPTN)
             SREL(Z038) .
  ENDUCL .
  SET BDY(ENMDLB) .                                             MODIFY
  UCLIN .
         ADD DLIBZONE(ENMDLB)                                   MODIFY
             RELATED(ENMTGT)                                    MODIFY
             OPTIONS(ENMOPTN)
             SREL(Z038) .
  ENDUCL .
```

Figure 5. Initialize SMP/E Zones (Part 1 of 4)

```
SET BDY(GLOBAL) .
UCLIN .
  ADD DDDEF  (SMPTLIB) UNIT(SYSDA) .                          MODIFY
  ADD DDDEF  (SMPSCDS) DATASET (CONN.SMPSCDS) SHR .            MODIFY
  ADD DDDEF  (SMPMTS) DATASET (CONN.SMPMTS) SHR .              MODIFY
  ADD DDDEF  (SMPPTS) DATASET (CONN.SMPPTS) SHR .              MODIFY
  ADD DDDEF  (SMPSTS) DATASET (CONN.SMPSTS) SHR .              MODIFY
  ADD DDDEF  (SMPLOG) DATASET (CONN.SMPLOG) MOD .              MODIFY
  ADD DDDEF  (SMPOUT) SYSOUT(*) .
  ADD DDDEF  (SMPRPT) SYSOUT(*) .
  ADD DDDEF  (SMPSNAP) SYSOUT(*) .
  ADD DDDEF  (SMPLIST) SYSOUT(*) .
  ADD DDDEF  (SYSPRINT) SYSOUT(*) .
  ADD DDDEF  (SYSUT1) CYL SPACE(10,5) UNIT(SYSDA) NEW DELETE .
  ADD DDDEF  (SYSUT2) CYL SPACE(10,5) UNIT(SYSDA) NEW DELETE .
  ADD DDDEF  (SYSUT3) CYL SPACE(10,5) UNIT(SYSDA) NEW DELETE .
  ADD DDDEF  (SYSUT4) CYL SPACE(10,5) UNIT(SYSDA) NEW DELETE .
  ADD DDDEF  (SMPWRK1) BLK(3120) SPACE(400,40) DIR(100)
                       UNIT(SYSDA) NEW DELETE .
  ADD DDDEF  (SMPWRK2) BLK(3120) SPACE(400,40) DIR(100)
                       UNIT(SYSDA) NEW DELETE .
  ADD DDDEF  (SMPWRK3) BLK(3120) SPACE(400,40) DIR(100)
                       UNIT(SYSDA) NEW DELETE .
  ADD DDDEF  (SMPWRK4) BLK(3120) SPACE(400,40) DIR(100)
                       UNIT(SYSDA) NEW DELETE .
  ADD DDDEF  (SMPWRK6) BLK(3120) SPACE(400,40) DIR(100)
                       UNIT(SYSDA) NEW DELETE .
ENDUCL .
```

Figure 5. Initialize SMP/E Zones (Part 2 of 4)

```
SET BDY(ENMTGT) .                                                MODIFY
UCLIN .
  ADD DDDEF  (SMPTLIB) UNIT(SYSDA) .                            MODIFY
  ADD DDDEF  (SMPSCDS) DATASET (CONN.SMPSCDS) SHR .              MODIFY
  ADD DDDEF  (SMPMTS) DATASET (CONN.SMPMTS) SHR .                MODIFY
  ADD DDDEF  (SMPPTS) DATASET (CONN.SMPPTS) SHR .                MODIFY
  ADD DDDEF  (SMPSTS) DATASET (CONN.SMPSTS) SHR .                MODIFY
  ADD DDDEF  (SMPLOG) DATASET (CONN.SMPLOG) MOD .                MODIFY
  ADD DDDEF  (SMPOUT) SYSOUT(*) .
  ADD DDDEF  (SMPRPT) SYSOUT(*) .
  ADD DDDEF  (SMPSNAP) SYSOUT(*) .
  ADD DDDEF  (SMPLIST) SYSOUT(*) .
  ADD DDDEF  (SYSPRINT) SYSOUT(*) .
  ADD DDDEF  (SYSUT1) CYL SPACE(10,5) UNIT(SYSDA) NEW DELETE .
  ADD DDDEF  (SYSUT2) CYL SPACE(10,5) UNIT(SYSDA) NEW DELETE .
  ADD DDDEF  (SYSUT3) CYL SPACE(10,5) UNIT(SYSDA) NEW DELETE .
  ADD DDDEF  (SYSUT4) CYL SPACE(10,5) UNIT(SYSDA) NEW DELETE .
  ADD DDDEF  (SMPWRK1) BLK(3120) SPACE(400,40) DIR(100)
                       UNIT(SYSDA) NEW DELETE .
  ADD DDDEF  (SMPWRK2) BLK(3120) SPACE(400,40) DIR(100)
                       UNIT(SYSDA) NEW DELETE .
  ADD DDDEF  (SMPWRK3) BLK(3120) SPACE(400,40) DIR(100)
                       UNIT(SYSDA) NEW DELETE .
  ADD DDDEF  (SMPWRK4) BLK(3120) SPACE(400,40) DIR(100)
                       UNIT(SYSDA) NEW DELETE .
  ADD DDDEF  (SMPWRK6) BLK(3120) SPACE(400,40) DIR(100)
                       UNIT(SYSDA) NEW DELETE .
  ADD DDDEF  (SYSLIB)   DATASET (SYS1.MACLIB) SHR .              MODIFY
  ADD DDDEF  (AENMINS0) DATASET (CONN.AENMINS0) SHR .            MODIFY
  ADD DDDEF  (AENMSRC0) DATASET (CONN.AENMSRC0) SHR .            MODIFY
  ADD DDDEF  (AENMMAC0) DATASET (CONN.AENMMAC0) SHR .            MODIFY
  ADD DDDEF  (AENMOBJ0) DATASET (CONN.AENMOBJ0) SHR .            MODIFY
  ADD DDDEF  (SENMINS0) DATASET (CONN.SENMINS0) SHR .            MODIFY
  ADD DDDEF  (SENMSRC0) DATASET (CONN.SENMSRC0) SHR .            MODIFY
  ADD DDDEF  (SENMMAC0) DATASET (CONN.SENMMAC0) SHR .            MODIFY
  ADD DDDEF  (SENMOBJ0) DATASET (CONN.SENMOBJ0) SHR .            MODIFY
ENDUCL .
```

Figure 5. Initialize SMP/E Zones (Part 3 of 4)

```
  SET BDY(ENMDLB) .                                              MODIFY
  UCLIN .
    ADD DDDEF  (SMPTLIB) UNIT(SYSDA) .                           MODIFY
    ADD DDDEF  (SMPSCDS) DATASET (CONN.SMPSCDS) SHR .            MODIFY
    ADD DDDEF  (SMPMTS) DATASET (CONN.SMPMTS) SHR .              MODIFY
    ADD DDDEF  (SMPPTS) DATASET (CONN.SMPPTS) SHR .              MODIFY
    ADD DDDEF  (SMPSTS) DATASET (CONN.SMPSTS) SHR .              MODIFY
    ADD DDDEF  (SMPLOG) DATASET (CONN.SMPLOG) MOD .              MODIFY
    ADD DDDEF  (SMPOUT) SYSOUT(*) .
    ADD DDDEF  (SMPRPT) SYSOUT(*) .
    ADD DDDEF  (SMPSNAP) SYSOUT(*) .
    ADD DDDEF  (SMPLIST) SYSOUT(*) .
    ADD DDDEF  (SYSPRINT) SYSOUT(*) .
    ADD DDDEF  (SYSUT1) CYL SPACE(10,5) UNIT(SYSDA) NEW DELETE .
    ADD DDDEF  (SYSUT2) CYL SPACE(10,5) UNIT(SYSDA) NEW DELETE .
    ADD DDDEF  (SYSUT3) CYL SPACE(10,5) UNIT(SYSDA) NEW DELETE .
    ADD DDDEF  (SYSUT4) CYL SPACE(10,5) UNIT(SYSDA) NEW DELETE .
    ADD DDDEF  (SMPWRK1) BLK(3120) SPACE(400,40) DIR(100)
                         UNIT(SYSDA) NEW DELETE .
    ADD DDDEF  (SMPWRK2) BLK(3120) SPACE(400,40) DIR(100)
                         UNIT(SYSDA) NEW DELETE .
    ADD DDDEF  (SMPWRK3) BLK(3120) SPACE(400,40) DIR(100)
                         UNIT(SYSDA) NEW DELETE .
    ADD DDDEF  (SMPWRK4) BLK(3120) SPACE(400,40) DIR(100)
                         UNIT(SYSDA) NEW DELETE .
    ADD DDDEF  (SMPWRK6) BLK(3120) SPACE(400,40) DIR(100)
                         UNIT(SYSDA) NEW DELETE .
    ADD DDDEF  (SYSLIB)   DATASET (SYS1.MACLIB) SHR .            MODIFY
    ADD DDDEF  (AENMINS0) DATASET (CONN.AENMINS0) SHR .          MODIFY
    ADD DDDEF  (AENMSRC0) DATASET (CONN.AENMSRC0) SHR .          MODIFY
    ADD DDDEF  (AENMMAC0) DATASET (CONN.AENMMAC0) SHR .          MODIFY
    ADD DDDEF  (AENMOBJ0) DATASET (CONN.AENMOBJ0) SHR .          MODIFY
    ADD DDDEF  (SENMINS0) DATASET (CONN.SENMINS0) SHR .          MODIFY
    ADD DDDEF  (SENMSRC0) DATASET (CONN.SENMSRC0) SHR .          MODIFY
    ADD DDDEF  (SENMMAC0) DATASET (CONN.SENMMAC0) SHR .          MODIFY
    ADD DDDEF  (SENMOBJ0) DATASET (CONN.SENMOBJ0) SHR .          MODIFY
  ENDUCL .
/*
//
```

Figure 5. Initialize SMP/E Zones (Part 4 of 4)

Step 4 - ENMAC04 - Unload and Install the MERVA Connection/ESA Distribution Tape with SMP/E

Now you can install MERVA Connection/ESA with SMP/E (sample job ENMAC04).

Job ENMAC04 will result in large output and high CPU usage. Ensure that appropriate execution classes and parameters are chosen.

When RACF is used in your system, and you let SMP/E allocate the temporary data sets for unloading the tape, you must instruct SMP/E to use an appropriate first qualifier for the data set names of the temporary libraries. (See job ENMAC03, parameter DSPREFIX).

```
//ENMAC04  JOB (ACCT),NAME,MSGCLASS=L,CLASS=Z                    MODIFY
//*************************************************************
//* SMP/E RECEIVE AND APPLY PROCESSING FOR THE PRODUCT
//* CONNECTION/ESA: H0HG330
//*
//* MODIFY THE UNIT PARAMETER OF DD CARD SMPPTFIN ACCORDING
//* TO YOUR INSTALLATION
//* -----------------------------------------------------------
//* PLEASE REMOVE COMMENTS FROM CONTROL CARDS BEFORE JOB START
//* -----------------------------------------------------------
//*************************************************************
//RCVAPP EXEC PGM=GIMSMP,REGION=4M,
//             PARM='CSI=CONN.SMPCSI.CSI'                        MODIFY
//SMPPTFIN DD DSN=SMPMCS,DISP=(OLD,KEEP),UNIT=TAPE,              MODIFY
//  VOL=SER=0HG330
//SMPHOLD  DD *
++NULL.
//SMPCNTL  DD *
SET BOUNDARY (GLOBAL).
RECEIVE SELECT(H0HG330).
SET BOUNDARY (ENMTGT) OPTIONS(ENMOPTN).                          MODIFY
APPLY SELECT(H0HG330).
/*
//
```

Figure 6. Unload and Install MERVA Connection/ESA with SMP/E

Step 5 - ENMAC05 - Prelink-edit and Link-edit the MERVA Connection/ESA Sample API Programs

MERVA Connection/ESA object modules require C/370 pre-linkage. The customer application may also require C/370 pre-linkage. This means that the application object module and the MERVA Connection/ESA object modules must be prelink-edited and then link-edited resolving all external references.

This is why MERVA Connection/ESA does not provide any load module to be used by a customer application. The load modules generated in this step are sample programs for demonstration purposes only.

So whenever any MERVA Connection/ESA object module was maintained, you have to prelink-edit and link-edit your affected load modules. Therefore you should extend job ENMAC05 with the appropriate JCL for prelink-editing and link-editing all your affected load modules.

The return code of the prelink-edit step is 4, the return code of the link-edit step has to be 0 for each of the sample programs.

```
//ENMAC05  JOB (ACCT),NAME,MSGCLASS=L,CLASS=Z                   MODIFY
//*************************************************************
//* THIS JOB PERFORMS THE C/370 PRE-LINKAGE AND CALLS THE
//* LINKAGE EDITOR HEWL FOR THE FINAL LINK OF THE
//* CONNECTION/ESA SAMPLE API LOAD MODULES.
//*
//* THE JOB MAY BE EXTENDED TO PRE-LINK AND LINK-EDIT YOUR
//* APPLICATION WITH THE CONNECTION/ESA OBJECT MODULES
//* CONTAINED IN THE SMP/E TARGET LIBRARY WITH THE LOW LEVEL
//* QUALIFIER SENMOBJ0.
//*
//* THE JOB MUST BE RUN AFTER MAINTENANCE OF ANY
//* CONNECTION/ESA OBJECT MODULE IN ORDER TO RELINK-EDIT
//* ALL APPLICABLE LOAD MODULES.
//*
//* PARAMETERS:
//*   CONN     HIGH LEVEL QUALIFIER(S) FOR SMP/E TARGET LIBRARIES
//*   CICS     CICS LOAD LIBRARY
//*   CCOMP    C/370 COMPILER MODULES
//*   CLIB     C/370 DYNAMIC LIBRARY
//*   CMSGS    C/370 COMPILER MESSAGES
//*   CSTUBS   C/370 LIBRARY STUBS
//*   COMLIB   COMMON DYNAMIC RUNTIME LIBRARY
//*   COMSTUBS COMMON RUNTIME LIBRARY STUBS
//*   UNIT     UNIT TYPE FOR WORKFILES
//*   SPACE    SPACE ALLOCATED FOR WORKFILES
//*   DCB      DCB FOR LRECL OF 80
//* -------------------------------------------------------------
//* PLEASE REMOVE COMMENTS FROM CONTROL CARDS BEFORE JOB START
//* -------------------------------------------------------------
//*************************************************************
```

Figure 7. Prelink-edit and Link-edit MERVA Connection/ESA Sample Programs (Part 1 of 3)

```
//LKCONN  PROC CONN='CONN',                                        MODIFY
//             CICS='CICS330.SDFHLOAD',                            MODIFY
//             CCOMP='EDC.V2R1M0.SEDCCOMP',                        MODIFY
//             CLIB='EDC.V2R1M0.SEDCLINK',                         MODIFY
//             CMSGS='EDC.V2R1M0.SEDCMSGS',                        MODIFY
//             CSTUBS='EDC.V2R1M0.SEDCBASE',                       MODIFY
//             COMLIB='PLI.V2R3M0.SIBMLINK',                       MODIFY
//             COMSTUBS='PLI.V2R3M0.SIBMBASE',                     MODIFY
//             UNIT=SYSDA,                                         MODIFY
//             SPACE='(32000,(30,30))',                            MODIFY
//             DCB='(RECFM=FB,LRECL=80,BLKSIZE=3200)'              MODIFY
//* ------------------------------------------------------------
//* PRE-LINKEDIT STEP
//* ------------------------------------------------------------
//PLKED     EXEC PGM=EDCPRLK,REGION=2048K
//STEPLIB    DD DSN=&CLIB.,DISP=SHR
//           DD DSN=&COMLIB.,DISP=SHR
//           DD DSN=&CCOMP.,DISP=SHR
//SYSMSGS    DD DSN=&CMSGS.(EDCPMSGE),DISP=SHR
//SYSLIB     DD DSN=&CONN..SENMOBJ0,DISP=SHR
//SYSOUT     DD SYSOUT=*
//SYSPRINT   DD SYSOUT=*
//SYSMOD     DD DSN=&&PLKSET,UNIT=&UNIT.,SPACE=&SPACE.,DCB=&DCB.,
//             DISP=(MOD,PASS)
//* ------------------------------------------------------------
//* LINKEDIT STEP
//* ------------------------------------------------------------
//LKED      EXEC PGM=HEWL,PARM='XREF,LIST,LET,RENT,AMODE=31,RMODE=ANY'
//SYSLIB     DD DSN=&CSTUBS.,DISP=SHR
//           DD DSN=&COMSTUBS.,DISP=SHR
//           DD DSN=&CICS.,DISP=SHR
//SYSPRINT   DD SYSOUT=*
//SYSUT1     DD UNIT=&UNIT.,SPACE=&SPACE.
//SYSLIN     DD DSN=*.PLKED.SYSMOD,DISP=(OLD,DELETE)
//           DD DDNAME=SYSIN
//SYSLMOD    DD DSN=&CONN..SENMLOD0,DISP=SHR
//         PEND
```

Figure 7. Prelink-edit and Link-edit MERVA Connection/ESA Sample Programs (Part 2 of 3)

```
//LKABEND   EXEC LKCONN
//PLKED.SYSIN DD *
   INCLUDE SYSLIB(ENMABEN)
   INCLUDE SYSLIB(ENMRAPI,ENMRPRF,ENMRUTL,ENMSNIL)
/*
//LKED.SYSIN DD *
   INCLUDE SYSLIB(DFHELII,DFHCPLC)
   ORDER DFHELII,CEESTART
   ENTRY CEESTART
   NAME ENMABEND(R)
/*
//LKSAMP    EXEC LKCONN
//PLKED.SYSIN DD *
   INCLUDE SYSLIB(ENMSAM)
   INCLUDE SYSLIB(ENMRAPI,ENMRPRF,ENMRUTL,ENMSSEC)
/*
//LKED.SYSIN DD *
   INCLUDE SYSLIB(DFHELII,DFHCPLC)
   ORDER DFHELII,CEESTART
   ENTRY CEESTART
   NAME ENMSAMP(R)
/*
//LKVERIF   EXEC LKCONN
//PLKED.SYSIN DD *
   INCLUDE SYSLIB(ENMVERI)
   INCLUDE SYSLIB(ENMRAPI,ENMRPRF,ENMRUTL,ENMSNIL)
/*
//LKED.SYSIN DD *
   INCLUDE SYSLIB(DFHELII,DFHCPLC)
   ORDER DFHELII,CEESTART
   ENTRY CEESTART
   NAME ENMVERIF(R)
/*
//
```

Figure 7. Prelink-edit and Link-edit MERVA Connection/ESA Sample Programs (Part 3 of 3)

Job ENMAC05 will result in large output. Ensure that appropriate execution classes and parameters are chosen.

Step 6 - ENMAC06 - Final SMP/E Installation Step of MERVA Connection/ESA

ACCEPT MERVA Connection/ESA to the distribution zone (member ENMAC06).

```
//ENMAC06  JOB (ACCT),NAME,MSGCLASS=L,CLASS=Z                    MODIFY
//*************************************************************
//* SMP/E ACCEPT PROCESSING FOR THE PRODUCT
//* CONNECTION/ESA: H0HG330
//* -----------------------------------------------------------
//* PLEASE REMOVE COMMENTS FROM CONTROL CARDS BEFORE JOB START
//* -----------------------------------------------------------
//*************************************************************
//ACCEPT EXEC PGM=GIMSMP,REGION=4M,
//             PARM='CSI=CONN.SMPCSI.CSI'                        MODIFY
//SMPHOLD  DD *
++NULL.
//SMPCNTL  DD *
SET BOUNDARY (ENMDLB) OPTIONS(ENMOPTN).                         MODIFY
ACCEPT SELECT(H0HG330).
/*
//
```

Figure 8. Update SMP/E Distribution Zone

Job ENMAC06 will result in large output and high CPU usage. Ensure that appropriate execution classes and parameters are chosen.

Install the MERVA Connection/ESA Data Sets

After successful installation of MERVA Connection/ESA from the distribution tape you have to allocate and initialize the following sequential data sets required by MERVA Connection/ESA:

- Connection profile
- Programmer's log
- Diagnosis log

Job ENMFILES in the library with the low level qualifier SENMINS0 allocates these data sets and initializes the connection profile with sample profile data.

```
//ENMFILES JOB (ACCT),NAME,MSGCLASS=L,CLASS=Z                    MODIFY
//*****************************************************************
//*   THIS JOB COMPRISES TWO STEPS IN PROCEDURE APISDS:
//*     ALLOCDS: CREATE CONNECTION/ESA API SEQUENTIAL DATASETS
//*     PROFILE: LOAD CONNECTION PROFILE DATASET
//*   -------------------------------------------------------
//*   PLEASE REMOVE COMMENTS FROM CONTROL CARDS BEFORE JOB START
//*   -------------------------------------------------------
//*****************************************************************
//APISDS PROC PRFX='CONN',UNIT=SYSDA                             MODIFY
//*****************************************************************
//* CREATE CONNECTION/ESA API SEQUENTIAL DATASETS
//*****************************************************************
//ALLOCDS  EXEC PGM=IEFBR14
//F1  DD DSN=&PRFX..CONNPROF,UNIT=&UNIT.,       CONNECTION PROFILE
//           SPACE=(80,(1,0),,CONTIG),
//           DCB=(DSORG=PS,RECFM=F,BLKSIZE=80,LRECL=80),
//           DISP=(NEW,CATLG)
//F2  DD DSN=&PRFX..PROGLOG,UNIT=&UNIT.,        PROGRAMMER'S LOG
//           SPACE=(136,(1000,100),,CONTIG),
//           DCB=(DSORG=PS,RECFM=V,BLKSIZE=136,LRECL=132),
//           DISP=(NEW,CATLG)
//F3  DD DSN=&PRFX..DIAGLOG,UNIT=&UNIT.,        DIAGNOSIS LOG
//           SPACE=(136,(500,50),,CONTIG),
//           DCB=(DSORG=PS,RECFM=V,BLKSIZE=136,LRECL=132),
//           DISP=(NEW,CATLG)
//*****************************************************************
//* LOAD CONNECTION PROFILE DATASET
//*****************************************************************
//PROFILE  EXEC PGM=IEBGENER,COND=(0,LT,ALLOCDS)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT2   DD DSN=&PRFX..CONNPROF,DISP=OLD      CONNECTION PROFILE
//       PEND
//*****************************************************************
//* CALL PROCEDURE APISDS, PROVIDE CONNECTION PROFILE DATASET RECORDS
//*****************************************************************
//       EXEC APISDS
//PROFILE.SYSUT1 DD *
4
PLOG
DLOG
M2API
ENMMICTL
ESA
/*
//
```

Figure 9. Allocate MERVA Connection/ESA Sequential Data Sets

MERVA Connection/ESA also requires a message integrity[1] control file, a VSAM data set. Job ENMMICTL in the library with the low level qualifier SENMINS0 allocates this data set.

1. The message integrity control file is used by the internal integrity processing of MERVA Connection/ESA, described in Figure 17 on page 51.

```
//ENMMICTL JOB (ACCT),NAME,MSGCLASS=L,CLASS=Z                    MODIFY
//*****************************************************************
//*  THIS JOB ALLOCATES THE CONNECTION/ESA MESSAGE INTEGRITY FILE
//*  ------------------------------------------------------------
//*  PLEASE REMOVE COMMENTS FROM CONTROL CARDS BEFORE JOB START
//*  ------------------------------------------------------------
//*****************************************************************
//DELDEF  EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  DELETE (CONN.MIPFILE) -                                       MODIFY
     CLUSTER PURGE NOERASE
  IF MAXCC = 8 THEN SET MAXCC = 0
  DEFINE  CLUSTER -
          (NAME(CONN.MIPFILE) -                                 MODIFY
          VOLUMES(US9008) -                                     MODIFY
          ERASE -
          SHAREOPTIONS ( 2 3 ) -
          KEYS(8 0) -
          RECSZ(12 12)) -
        DATA (RECORDS (20 10)) -
        INDEX (IMBED)
/*
//
```

Figure 10. Allocate MERVA Connection/ESA Message Integrity Control File

Chapter 4. Customization and Network Definitions

This chapter describes how, after you have installed MERVA Connection/ESA, you can do the customization on the S/390, the OS/2 system, and the RS/6000.

Customization on the S/390

The customization of MERVA Connection/ESA on the S/390 requires appropriate definitions in the following areas:

- VTAM
- CICS

Customizing VTAM

The connection between the S/390 and the OS/2 system or between the S/390 and the RS/6000 must be an LU 6.2 session. Make the following definitions:

- Physical Unit (PU)

Define the OS/2 or the AIX system as a physical unit by means of the VTAM macro PU. This enables the OS/2 or the AIX system to be activated as a PU 2.1 node in the SNA network.

- Local node ID

If your OS/2 system resides on a Token Ring network and you have defined the IDNUM parameter in the PU macro, the IDNUM value specifies your local node ID defined in the Communications Server.

- XID_NODE_ID

If your RS/6000 resides on a Token Ring network and you have defined the IDNUM parameter in the PU macro, the IDNUM value specifies your xid_node_id defined in the control point of the SNA Server.

- LAN destination address

The LAN destination address must be defined in the network control program (NCP) of the control unit used. This parameter is only necessary when the connection is established via a Token Ring network.

- SNA network ID

The SNA network ID must be specified in the start option ”NETID=...’’ when VTAM is started.

- Partner node name

The partner node name is not important if you specify the PU using the local node ID (IDNUM parameter).

- LU name of CICS (CICS APPLID)

You specify the LU name of CICS when you define the CICS application by means of VTAM macro APPL.

- LU name of the OS/2 system

You specify the LU name of the OS/2 system by means of VTAM macro LU.

The LU 6.2 session is established between the CICS APPLID and the OS/2 local LU defined as an independent LU in the OS/2 system LU macro.

- LU name of the RS/6000

You specify the LU name of the RS/6000 by means of VTAM macro LU.

The LU 6.2 session is established between the CICS APPLID and the AIX local LU defined as an independent LU in the SNA Server local LU profile.

- Logmode name

You specify the appropriate entries in the Logon mode table by means of VTAM macro MODEENT.

In “VTAM Definitions: OS/2 System Connected to a Token Ring Network” on page 75 and in “VTAM Definitions: OS/2 System Connected by an SDLC Line” on page 76 you find examples for many of these definitions.

Customizing CICS

You make definitions in the following resources:

- CSD (CICS System Definition)
- DCT (Destination Control Table)
- SIT (System Initialization Table)

You find the members for the CSD and the DCT in the library with the low level qualifier SENMMAC0. The name of the CSD member is ENMCSD, the name of the DCT member is ENMDCT.

- **CSD**

  Run CICS/ESA utility DFHCSDUP when you make these definitions:

  – Program names (DEFINE PROGRAM ...)

  ENMVERIF Installation verification

  ENMABEND Installation verification abnormal termination exit

  ENMSAMP Sample for loading telex messages

  You should check whether the C/370 runtime library is already defined to CICS in your installation. If not, you could define the runtime library as follows:

  ```
  DEFINE PROGRAM(EDCXV) GROUP(ENMGROUP) LANGUAGE(ASSEMBLER)
  ```

  – Transaction codes (DEFINE TRANSACTION ...)

  ENM2 Installation verification

  SAMP Sample for loading telex messages

  – File definition (DEFINE FILE(ENMMICTL) ...)

  ENMMICTL DD-name of the message integrity control file in your CICS startup job.

  Name of the message integrity control file in the MERVA Connection/ESA profile.

  – Network definitions

  Session profile
  DEFINE PROFILE(ENMPROF) ...

Connection DEFINE CONNECTION(RAPI) ...

Sessions DEFINE SESSIONS(C3RAPI) ...

Partner DEFINE PARTNER(M2API) ...

M2API Name of CSI[2] object in the MERVA Connection/ESA profile.

The following table shows the relationship of parameters specified in the CSD network definitions to other parameters.

In DEFINE ...	Parameter ...	Must correspond to ...
PROFILE(ENMPROF)	MODENAME(CICSISC)	VTAM Logon Mode table entry
CONNECTION(RAPI)	NETNAME(FD577300)	LU name of the OS/2 system or LU name of the RS/6000
SESSIONS(C3RAPI)	CONNECTION(RAPI)	DEFINE CONNECTION(RAPI)
	MODEMNAME(CICSISC)	VTAM Logon Mode table entry
PARTNER(M2API)	NETNAME(FD577300)	DEFINE CONNECTION(RAPI NETNAME(...)
	PROFILE(ENMPROF)	DEFINE PROFILE(ENMPROF)
	TPNAME(MERVA2)	Transaction program name of the Remote MERVA API Server
	or	in MERVA OS/2 V3
	TPNAME(ENMRAS) PARNTER(M2API)	or
		Transaction program name of the Remote MERVA API Server in MERVA AIX
		MERVA Connection/ESA profile line 4

In “CICS Definitions” on page 77 you see how the network definitions are used as input for DFHCSDUP. Modify the definitions contained in member ENMCSD according to your needs and run DFHCSDUP.

- **DCT**

  Member ENMDCT contains the definitions for the following extrapartition destinations. You may include them in your DCT using ENMDCT as a copy book. Then you assemble and link-edit the DCT.

  – MERVA Connection/ESA profile
  – Programmer’s log
  – Diagnosis log

  The MERVA Connection/ESA profile is then identified by these names:

  ENMCPROF DD-name of the MERVA Connection/ESA profile in your CICS startup job.

  PROF Name of the MERVA Connection/ESA profile in your application when you select the profile.

2. Communication Side Information - object in CPI Communications containing initialization parameters. These are, for example:
   - The name of the partner program (such as the Remote MERVA API Server) with which a program can establish a conversation
   - The name of the logical unit (LU) at the partner program’s node, which CPI Communications requires to establish a conversation.

The programmer's log is then identified by these names:

ENMPLOG DD-name of the programmer's log in your CICS startup job.

PLOG Name of the programmer's log in the MERVA Connection/ESA profile.

The diagnosis log is then identified by these names:

ENMDLOG DD-name of the diagnosis log in your CICS startup job.

DLOG Name of the diagnosis log in the MERVA Connection/ESA profile.

- **SIT**

  You can make the following definitions immediately in your CICS startup job. If you prefer to make them in the SIT then you have to assemble and link-edit the SIT.

  – ISC=YES

  – DCT=xx

    where xx is the suffix of your DCT.

After all these definitions, the relevant JCL statements for MERVA Connection/ESA in a CICS/ESA Version 3 Release 3.0 startup job may look like this (note that &PRFX represents the high level qualifier(s) you chose when you allocated the data sets):

```
//CICS    EXEC PGM=DFHSIP,REGION=32M,PARM=SYSIN
//STEPLIB DD DSN=CICS330.SDFHAUTH,DISP=SHR   CONTAINS EDCCICS (C-CICS INTERFACE)
//DFHRPL  DD DSN=&PRFX.SENMLOD0,DISP=SHR     MERVA CONNECTION/ESA LOAD LIBRARY
//        DD DSN=XYZ.SEDCLINK,DISP=SHR       CONTAINS EDCXV (RUNTIME LIBRARY)
//        DD DSN=CICS330.SDFHLOAD,DISP=SHR
          ..............................
          .. further data sets
          ..............................
//SYSIN   DD    *
SIT=6$,
APPLID=I40AC388,                          DEFINED IN VTAM MACRO APPL
GRPLIST=XXXLIST,                          CONTAINS MERVA CONNECTION/ESA GROUP ENMGROUP
ISC=YES,
DCT=xx,
          ..............................
          .. further SIT overrides
          ..............................
 .END
/*
//*************************************************************
//*      MERVA CONNECTION/ESA DATA SETS
//*************************************************************
//ENMCPROF DD DSN=&PRFX.CONNPROF,DISP=SHR    CONNECTION PROFILE
//ENMPLOG  DD DSN=&PRFX.PROGLOG,DISP=SHR     PROGRAMMER'S LOG
//ENMDLOG  DD DSN=&PRFX.DIAGLOG,DISP=SHR     DIAGNOSIS LOG
//ENMMICTL DD DSN=&PRFX.MIPFILE,DISP=SHR     MESSAGE INTEGRITY CONTROL
//*************************************************************
//*      TRANSIENT DATA QUEUES, CCSO ALIAS COUT REQUIRED AT LEAST
//*************************************************************
//COUT     DD SYSOUT=*,DCB=(DSORG=PS,RECFM=V,BLKSIZE=137)   C/370 STDOUT
//CCSE     DD SYSOUT=*,DCB=(DSORG=PS,RECFM=V,BLKSIZE=137)   C/370 STDERR
//CCPI     DD SYSOUT=*,DCB=(DSORG=PS,RECFM=V,BLKSIZE=137)   CPI-C MESSAGE LOG
//CSRL     DD SYSOUT=*,DCB=(DSORG=PS,RECFM=V,BLKSIZE=137)   PARTNER RESOURCE LOG
           ..............................
           .. further data sets
           ..............................
//
```

Settings in the MERVA Connection/ESA Profile

You must provide MERVA Connection/ESA with information about logging, network partner, and internal customization parameters. This information is provided in the MERVA Connection/ESA profile. The profile must be a sequential data set and defined as an extrapartition destination in CICS/ESA. With the API call ENMSetProfile (described in “Structure of the MERVA API Program on the S/390” on page 37), you can tell MERVA Connection/ESA the symbolic name of the extrapartition destination which represents the profile. You specify this name in the CICS/ESA DCT using the macro DFHDCT TYPE=EXTRA,DESTID=`name`.

After successful execution of job ENMFILES (refer to “Install the MERVA Connection/ESA Data Sets” on page 21), the profile data set exists and contains sample profile data. Note that the data must begin in column one.

The following table shows the format of a MERVA Connection/ESA profile.

Line	Information	Sample
1	Logging level	4
2	Name of programmer’s log	PLOG
3	Name of diagnosis log	DLOG
4	Name of CSI object	M2API
5	Name of message integrity control file	ENMMICTL
6	System type	ESA

Note: Concurrently running API programs can use the same profile or may refer to different profiles. In each of the profiles, you may specify different logging files (lines 2 and 3 of the profile) and different message integrity control file names (line 5 of the profile).

Network Definitions on the OS/2 System

On the OS/2 system, local definitions, link definitions, and LU definitions must be made. These are not described here, but a sample configuration file for the Communications Server, called ES9000, is delivered on the MERVA OS/2 V3 Samples Diskette. The sample configuration file is configured for a Token Ring connection between an OS/2 system and an S/390. The definitions in the configuration file conform to the definitions given in the sample definitions for the S/390.

To adjust the sample profile to your environment, change the Token Ring adapter addresses and the LU names.

The Remote MERVA API Server program must be defined as a transaction program. The name of the delivered Remote MERVA API Server program is enmotpi.exe. The full path name is:

x:\MERVA2\BASE\ENMOTPI.EXE

where *x* is the drive where MERVA OS/2 V3 is installed.

3. You find more information on the logging levels in “Programmer’s Log” on page 71.

Network Definitions on the RS/6000

On the RS/6000 the following definitions must be done:

- Control Point
- Links
- Sessions
  - LU 6.2 Local LU
  - LU 6.2 Partner LU
  - Mode
  - Transaction Program Name

These are described in the definitions given in “Appendix B. Sample Network Definitions for the RS/6000” on page 79.

To adjust the sample profile to your environment, do the following:

- Enter **smitty sna**
- Select **Configure SNA Profiles**
- Select **Advanced Configuration**

Starting from the Advanced Configuration window, perform the following steps:

- Select **Control Point**
- Select **Change/Show a Profile**

Figure 11 shows a sample control point profile.

```
                    Change/Show Control Point Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Profile name                                        node_cp
  XID node ID                                        [071fea1d]
  Network name                                       [DEIBMFD]
  Control Point (CP) name                            [FDA71D]
  Control Point alias                                [FDA71D]
  Control Point type                                  appn_end_node           +
  Maximum number of cached routing trees             [500]                    #
  Maximum number of nodes in the TRS database        [500]                    #
  Route addition resistance                          [128]                    #

  Comments                                           [CP for ESA Connection >



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 11. Change/Show Control Point Profile

Change the *XID node ID* and the *Network name*.

- Select **Links**
- Select **Token Ring**
- Select **Token Ring Link Station**

- Select **Change/Show a Profile**
- Select **tokesa**

Figure 12 shows a sample token ring link station profile.

```
                    Change/Show Token Ring Link Station Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                 [Entry Fields]
  Current profile name                               tokesa
  New profile name                                   []
  Use Control Point's XID node ID?                    yes                          +
     If no, XID node ID                              [*]
* SNA DLC Profile name                              [tok0]                        +
  Stop link station on inactivity?                    no                           +
     If yes, Inactivity time-out (0-10 minutes)      [0]                           #
  LU address registration?                            no                           +
     If yes,
        LU Address Registration Profile name         []                           +
  Trace link?                                         no                           +
     If yes, Trace size                               long                         +

  Adjacent Node Address Parameters
     Access routing                                   link_address                 +
     If link_name, Remote link name                  []
     If link_address,
        Remote link address                          [400010000008]                X
        Remote SAP address (02-fa)                   [04]                          X
[MORE...]

F1=Help              F2=Refresh            F3=Cancel             F4=List
F5=Reset             F6=Command            F7=Edit               F8=Image
F9=Shell             F10=Exit              Enter=Do
```

Figure 12. Change/Show Token Ring Link Station Profile

Change the values of the *Remote link address* and the *Remote SAP address*.

- Select **Sessions**
- Select **LU 6.2**
- Select **LU 6.2 Local LU**
- Select **Change/Show a Profile**
- Select **MERESALLUP**

Figure 13 shows a sample LU 6.2 local LU profile.

```
                        Change/Show LU 6.2 Local LU Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Current profile name                                MERESALLUP
  New profile name                                   []
  Local LU name                                      [FDA71D00]
  Local LU alias                                     [FDA71D00]
  Local LU is dependent?                              no                                +
     If yes,
        Local LU address (1-255)                     []                                 #
        System services control point
           (SSCP) ID (*, 0-65535)                    [*]
        Link Station Profile name                    []                                +
  Conversation Security Access List Profile name     []
  Recovery resource manager (RRM) enabled?            no                                +

  Comments                                           [RemAPI from a CICS/ESA system]




F1=Help              F2=Refresh           F3=Cancel            F4=List
F5=Reset             F6=Command           F7=Edit              F8=Image
F9=Shell             F10=Exit             Enter=Do
```

Figure 13. Change/Show LU 6.2 Local LU Profile

Adapt the *Local LU name* and the *Local LU alias* to your needs.

- Select **Sessions**
- Select **LU 6.2**
- Select **LU 6.2 Partner LU**
- Select **Change/Show a Profile**
- Select **ESAPLUP**

Figure 14 shows a sample LU 6.2 partner LU profile.

```
                        Change/Show LU 6.2 Partner LU Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Current profile name                                ESAPLUP
  New profile name                                   []
  Fully qualified partner LU name                    [DEIBMID.I40AC388]
  Partner LU alias                                   [I40AC388]
  Parallel sessions supported?                        yes                          +
  Session security supported?                         no                           +
  Conversation security level                         already_verified             +

  Comments                                           [RemAPI: CICS/ESA system]









F1=Help              F2=Refresh           F3=Cancel            F4=List
F5=Reset             F6=Command           F7=Edit              F8=Image
F9=Shell             F10=Exit             Enter=Do
```

Figure 14. Change/Show LU 6.2 Partner LU Profile

Change the *Fully qualified partner LU name* and the *Partner LU alias*. The *Fully qualified partner LU name* consists of the network name and the CICS application ID separated by a period. It defines your partner CICS/ESA system.

- Select **Sessions**
- Select **LU 6.2**
- Select **LU 6.2 Mode**
- Select **Change/Show a Profile**
- Select **CICSISC**

You can use the CICSISC profile without changes.

- Select **Sessions**
- Select **LU 6.2**
- Select **LU 6.2 Transaction Program Name (TPN)**
- Select **Change/Show a Profile**
- Select **ENMRAS**

Figure 15 shows a sample LU 6.2 TPN profile.

```
                          Change/Show LU 6.2 TPN Profile

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                [Entry Fields]
  Current profile name                               ENMRAS
  New profile name                                   []
  Transaction program name (TPN)                     [ENMRAS]
  Transaction program name (TPN) is in hexadecimal?   no                              +
  PIP data?                                           no                              +
     If yes, Subfields (0-99)                        [0]                               #
  Use command line parameters?                        yes                             +
  Command line parameters                            [trace]
  Conversation type                                   mapped                          +
  Sync level                                          confirm                         +
  Resource security level                             none                            +
     If access, Resource Security Access List Prof.  []
  Full path to TP executable                         [/home/merva1/ipc/enmtpi.cmd]
  Multiple instances supported?                       no                              +
  User ID                                            [210]                             #
  Server synonym name                                [ENMRASRV]
  Restart action                                      once                            +
  Communication type                                  signals                         +
     If IPC, Communication IPC queue key             [0]                               #
  Time out Attaches?                                  yes                             +
    If yes, time-out value (0-3600 seconds)          [60]                              #
  Standard input file/device                         [/dev/null]
  Standard output file/device                        [/dev/null]
  Standard error file/device                         [/dev/null]

  Comments                                           [RemAPI server program]
[BOTTOM]

F1=Help              F2=Refresh           F3=Cancel            F4=List
F5=Reset             F6=Command           F7=Edit              F8=Image
F9=Shell             F10=Exit             Enter=Do
```

Figure 15. Change/Show LU 6.2 TPN Profile

The *Transaction program name* must correspond with the name entered in the CSI object on the remote application side.

Adapt the *Full path to TP executable* parameter to your needs. This parameter identifies your MERVA instance. Your MERVA instance is defined as selected in the Create MERVA AIX instance step described in the *MERVA AIX Installation and Customization Guide*.

In this example the name of the MERVA instance is *merva1*.

Adapt the *User ID* to your needs. For example, enter a user ID that has the necessary MERVA API rights. Dependent of the entered user ID, you should specify the appropriate *Standard input file/device*, *Standard output file/device* and *Standard error file/device*.

Chapter 5. Verifying Correct Installation and Customization

To verify that the installation and customization of MERVA Connection/ESA was successful, run the sample program ENMVERIF. The transaction code for ENMVERIF is ENM2.

Before you can run this program, the user ID SAMPLE with the password SAMPLE1 has to be defined in MERVA OS/2 V3 or on MERVA AIX. This user ID must be approved for application programs.

The program also checks that the queue API_IN has been customized. API_IN will be used in the sample program ENMSAMP which you should run after the installation was successfully verified. The transaction code for ENMSAMP is SAMP. For details on how ENMSAMP works, look into its program header in the source code contained in the library with the low level qualifier SENMSRC0.

You start both sample programs by entering the appropriate transaction code at the CICS terminal.

After you have entered the transaction code ENM2, the following output should appear on the screen:

```
Transaction ENM2 has started...
Profile name PROF specified
APPC Conversation named ENM2 is up
Program attached to MERVA
Queue API_IN exists
Program detached from MERVA
APPC Conversation successfully terminated
Transaction ENM2 has ended
```

If the output could not be displayed at the terminal, it is printed at the extrapartition destination COUT (or any other destination in your installation which represents the C/370 output device `stdout`).

In addition, there is logging information available due to the logging level 4 in the delivered sample profile PROF. You can inspect the logging data in the programmer's log contained in the data set with the low level qualifier PROGLOG. In case of errors, the diagnosis log which is represented by the data set with the low level qualifier DIAGLOG, shows more detailed information.

Chapter 6. The Application Programming Interface

The following description of the API is based on the description in *MERVA OS/2 V3 Application Programming* and *MERVA AIX Application Programming*. This chapter describes only the differences between MERVA API programming on the S/390 and MERVA OS/2 V3 API programming on the OS/2 system or MERVA AIX API programming on the RS/6000.

Structure of the MERVA API Program on the S/390

One major task of the MERVA API program on the S/390 is that it must call functions that deal with connecting and disconnecting to and from the OS/2 system or the RS/6000:

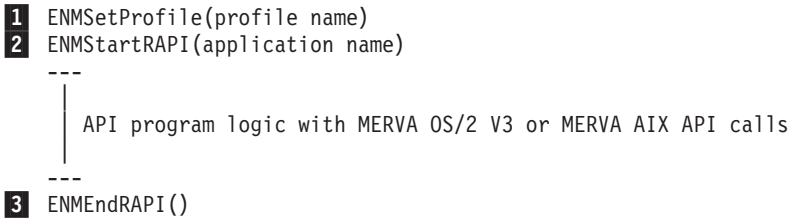

Figure 16. MERVA API Program Structure on the S/390

1 Before the API functions can be called, the Remote MERVA API Client on the S/390 must be initialized by calling the function ENMSetProfile. This function tells the Remote MERVA API Client the name of the profile. The profile is described in “Settings in the MERVA Connection/ESA Profile” on page 29.

2 After having set the profile name, the connection to the Remote MERVA API Server on MERVA OS/2 V3 or MERVA AIX can be established. To do this, call the function ENMStartRAPI. When this function is called, the Remote MERVA API Client is initialized and the network connection to the is established.

After the ENMStartRAPI call, the API functions can be called as if the program ran locally on a MERVA OS/2 V3 or MERVA AIX machine.

3 Before terminating the program, the connection to the Remote MERVA API Server must be released by calling the function ENMEndRAPI. It is important to call this function even if an error occurs in the API program, otherwise, the Remote MERVA API Server misses the termination and is not ready to receive the next connection request when the API program is started again.

C Language Data Types

The member ENM4RAPI of the library with the low level qualifier SENMMAC0 contains the data types and prototypes of the MERVA API functions. When compiling a MERVA OS/2 V3 API program locally on the OS/2 system or a MERVA AIX API program locally on the RS/6000, the file enmoapi.h is included. When compiling a MERVA Connection/ESA API program on the S/390, member ENM4RAPI is included as file enm4rapi.h instead.

ENM4RAPI contains some used data types, which are used to describe the API calls covered in this book. Their meaning is as follows:

Type	Definition
USHORT	unsigned short
UCHAR	unsigned char
PUCHAR	unsigned char*
PUSHORT	unsigned short*
ULONG	unsigned long
PULONG	unsigned long*

Additional Functions

MERVA Connection/ESA provides more API calls than the MERVA OS/2 V3 and MERVA AIX API. They are divided into the following categories:

- Functions for starting and ending the conversation
- Functions enabling the API program to be triggered by MERVA OS/2 V3 alarms
- Functions for error handling.

Starting and Ending the Conversation

If you want that the API program starts and ends the conversation between theRemote MERVA API Client and the Remote MERVA API Server, use the following functions:

- ENMSetProfile - Select a Profile
- ENMStartRAPI - Establish Connection to MERVA OS/2 V3 or MERVA AIX
- ENMRestartRAPI - Reconnect Remote API Program to MERVA OS/2 V3 or MERVA AIX
- ENMEndRAPI - Disconnect from MERVA OS/2 V3 or MERVA AIX.

Each function is described in the following.

ENMSetProfile - Select a Profile

Specify the name of the profile you want to use.

C Definition

```
void ENMSetProfile (PUCHAR pucProfileName);
```

Parameter Description

The following parameter is required:

pucProfileName(PUCHAR)

Pointer to a null-terminated string with a maximum length of 4 characters. The profile name is defined in the CICS/ESA DCT by macro DFHDCT TYPE=EXTRA,DESTID=name.

Note: If several API programs run concurrently, each may use the same profile name.

Remarks

The format and contents of the profile file are described in “Settings in the MERVA Connection/ESA Profile” on page 29.

C Language Example:

```
#include "enm4rapi.h"

ENMSetProfile ("PROF");
```

ENMStartRAPI - Establish Connection to MERVA OS/2 V3 or MERVA AIX

C Definition

```
USHORT ENMStartRAPI ( PUCHAR pucApplicationName );
```

Parameter Description

The following parameters are required:

- **retCode**(USHORT) - output

Code	Meaning
2	An internal error has occurred. The API program receives further information by calling the function ENMGetReason, described in “Handling Errors” on page 48. The reason code is also written to the diagnosis log on the S/390 (see “Chapter 11. Diagnosis Information” on page 71). If it is an internal error of the MERVA OS/2 V3 or MERVA AIX API, the reason code is zero.

- **pucApplicationName**(PUCHAR) - input

  A pointer to a null-terminated string of up to 8 characters. This name is registered by the Remote MERVA API Server.

  Note: If several API programs run concurrently, each must use a different name. For example, the transaction code can be used as a unique name.

Remarks

This call establishes the APPC conversation with MERVA OS/2 V3 or MERVA AIX (Remote MERVA API Server) and initializes internal buffers and variables. After this function was called, the program must not end without calling ENMEndRAPI.

C Language Example

```
#include "enm4rapi.h"

USHORT rc = 0;

if ((rc = ENMStartRAPI ( "APPLID" )) == 0 )
    puts("APPC Conversation is up\n");
else
    puts("Error in ENMStartRAPI");
```

ENMRestartRAPI - Reconnect Remote API Program to MERVA OS/2 V3 or MERVA AIX

C Definition

```
USHORT ENMRestartRAPI ( PUCHAR pucApplicationName );
```

Parameter Description

The following parameters are required:

- **retCode**(USHORT) - output

Code	Meaning
2	An internal error has occurred. The API program receives further information by calling the function ENMGetReason, described in “Handling Errors” on page 48. The reason code is also written to the diagnosis log on the S/390 (see “Chapter 11. Diagnosis Information” on page 71). If it is an internal error of the MERVA OS/2 V3 or MERVA AIX API, the reason code is zero.

- **pucApplicationName**(PUCHAR) - input

  A pointer to a null-terminated string of up to 8 characters. This name is registered by the Remote MERVA API Server.

  Note: If several API programs run concurrently, each must use a different name. For example, the transaction code can be used as a unique name.

Remarks

If the connection is established with this call instead of ENMStartRAPI, the resynchronization described in Figure 17 on page 51 is provided for the following API calls:

- ENMAdd
- ENMDelete
- ENMPut
- ENMRouteAdd
- ENMRoutePut

If the connection was not interrupted within the critical time period in a previous session, this call has the same functions as ENMStartRAPI. Therefore, you can also use it if the previous connection did not end abnormally.

C Language Example

```
#include "enm4rapi.h"

USHORT rc = 0;

if ((rc = ENMRestartRAPI ( "APPLID" )) == 0 )
    puts("APPC Conversation is up\n");
else
    puts("Error in ENMRestartRAPI");
```

ENMEndRAPI - Disconnect from MERVA OS/2 V3 or MERVA AIX

C Definition

```
USHORT ENMEndRAPI ( void );
```

Parameter Description

The following parameter is required:

retCode(USHORT) - output

Code	Meaning
2	An internal error has occurred. The API program receives further information by calling the function ENMGetReason, described in “Handling Errors” on page 48. The reason code is also written to the diagnosis log on the S/390 (see “Chapter 11. Diagnosis Information” on page 71). If it is an internal error of the MERVA OS/2 V3 or MERVA AIX API, the reason code is zero.

Remarks

The APPC conversation to MERVA OS/2 V3 or MERVA AIX is terminated.

C Language Example

```
#include "enm4rapi.h"

USHORT rc = 0;

if ((rc = ENMEndRAPI ()) == 0 )
    puts("APPC Conversation successfully terminated\n");
else
    puts("Error in ENMEndRAPI");
```

Functions Enabling the API Program to Be Triggered

If you want that the API program is triggered by MERVA OS/2 V3 or MERVA AIX alarms, use the following functions:

- ENMWaitSemList - Wait for a List of Semaphores
- ENMCloseSem - Close a Semaphore
- ENMSetSem - Set a Semaphore
- ENMClearSem - Clear a Semaphore
- ENMCreateSem - Create a Semaphore
- ENMOpen - Open a Semaphore.

The semaphores reside on the OS/2 system or RS/6000 system. A semaphore is cleared when an alarm is activated. An alarm is activated when a message enters the associated queue.

Each function is described in the following.

ENMWaitSemList - Wait for a List of Semaphores

This call blocks the current process until one of the specified OS/2 or AIX semaphores is cleared. It allows the API program to wait for a list of up to 16 semaphores and up to 16 different MERVA OS/2 V3 or MERVA AIX alarms.

C Definition

```
USHORT ENMWaitSemList(PUSHORT Index,
                      ULONG   timeout,
                      ULONG   SemHandle,
                              ...,
                     (ULONG)  0);[4]
```

Parameter Description

The following parameters are required:

- **retCode**(USHORT) - output

Code	Meaning
2	An internal error has occurred. The API program receives further information by calling the function ENMGetReason, described in “Handling Errors” on page 48. The reason code is also written to the diagnosis log on the S/390 (see “Chapter 11. Diagnosis Information” on page 71). If it is an internal error of the MERVA OS/2 V3 or MERVA AIX API, the reason code is zero.
121	No semaphore is cleared. The timeout was reached.
Others	See the description of OS/2 system call DosMuxSemWait in the OS/2 online help or see the description of the ENMWaitSemList function in the book *MERVA AIX Application Programming*.

- **Index**(PUSHORT) - output

  In this parameter, ENMWaitSemList returns an index (0..15) that tells you which of the semaphores is cleared.

- **timeout**(ULONG) - input

Code	Meaning
-1	Wait indefinitely for a semaphore to be cleared.
0	Return immediately.
>1	Wait the indicated number of milliseconds for a semaphore to be cleared before resuming execution.

- **SemHandle**(ULONG) - input

  Up to 16 semaphore handles that were created by the calls of ENMCreateSem or ENMOpenSem.

- **(ULONG)0** - input

  This parameter terminates the list of semaphores. Its value must be zero and a 4-byte value. If the parameter is missing, ENMWaitSemList is not able to recognize the end of the semaphore list.

4. The last parameter ((ULONG)0) is not part of the C function prototype. It is only mentioned here to show that the list of SemHandle parameters must be terminated by the value 0 (4 bytes).

C Language Example

```
/*
   If a connection to MERVA OS/2 V3 should be used,
   use the following define-statements:
*/
#define TRIGGER "\\SEM\\SAMPLE2"
#define STOP    "\\SEM\\STOP.SMP"
/*
   If a connection to MERVA AIX should be used,
   use the following define-statements:
*/
/*
  #define TRIGGER "SAMPLE2"
  #define STOP    "STOP.SMP"
*/
#include "enm4rapi.h"

   USHORT      rc = 0;
   ULONG       SemTrigger;
   ULONG       SemStop;
   USHORT      Index = 0;

   if ((rc = ENMCreateSem (&SemStop, STOP)) == 0)
      if ((rc = ENMCreateSem (&SemTrigger, TRIGGER)) == 0)
         if (( rc = ENMSetSem (SemStop)) == 0)
            if ((rc = ENMSetSem(SemTrigger)) == 0)
               rc = ENMWaitSemList(&Index, -1L,
                                   SemStop,
                                   SemTrigger,
                                   (ULONG)0);
```

ENMCloseSem - Close a Semaphore

This call closes an OS/2 or AIX semaphore obtained with an ENMCreateSem or ENMOpenSem call.

C Definition

```
USHORT ENMCloseSem (ULONG SemHandle);
```

Parameter Description

The following parameters are required:

- **retCode**(USHORT) - output

Code	Meaning
2	An internal error has occurred. The API program receives further information by calling the function ENMGetReason, described in “Handling Errors” on page 48. The reason code is also written to the diagnosis log on the S/390 (see “Chapter 11. Diagnosis Information” on page 71). If it is an internal error of the MERVA OS/2 V3 or MERVA AIX API, the reason code is zero.
Others	See the description of the OS/2 system call DosCloseSem in the OS/2 online help or see the description of the ENMCloseSem function in the book *MERVA AIX Application Programming*.

- **SemHandle**(ULONG) - input

  Generated by ENMCreateSem or ENMOpenSem.

C Language Example

```
/*
   If a connection to MERVA OS/2 V3 should be used,
   use the following define-statements:
*/
#define TRIGGER "\\SEM\\SAMPLE2"
/*
   If a connection to MERVA AIX should be used,
   use the following define-statements:
*/
/*
  #define TRIGGER "SAMPLE2"
*/
#include "enm4rapi.h"

   USHORT      rc = 0;
   ULONG       SemTrigger;

   if ((rc = ENMCreateSem (&SemTrigger, TRIGGER)) == 0)
      rc = ENMCloseSem (SemTrigger);
```

ENMSetSem - Set a Semaphore

C Definition

```
USHORT ENMSetSem (ULONG SemHandle);
```

Parameter Description

The following parameters are required:

- **retCode**(USHORT) - output

Code	Meaning
2	An internal error has occurred. The API program receives further information by calling the function ENMGetReason, described in “Handling Errors” on page 48. The reason code is also written to the diagnosis log on the S/390 (see “Chapter 11. Diagnosis Information” on page 71). If it is an internal error of the MERVA OS/2 V3 or MERVA AIX API, the reason code is zero.
Others	See the description of the OS/2 system call DosSemSet in the OS/2 online help or see the description of the ENMSetSem function in the book *MERVA AIX Application Programming*.

- **SemHandle**(ULONG) - input

  Generated by ENMCreateSem or ENMOpenSem.

Remarks

ENMSetSem sets a semaphore unconditionally. For MERVA OS/2 V3 or MERVA AIX this means that the semaphore can be cleared by raising an alarm.

C Language Example

```
/*
   If a connection to MERVA OS/2 V3 should be used,
   use the following define-statements:
*/
#define TRIGGER "\\SEM\\SAMPLE2"
/*
```

```
   If a connection to MERVA AIX should be used,
   use the following define-statements:
*/
/*
  #define TRIGGER "SAMPLE2"
*/
#include "enm4rapi.h"

   USHORT      rc = 0;
   ULONG       SemTrigger;

   if ((rc = ENMCreateSem (&SemTrigger, TRIGGER)) == 0)
      rc = ENMSetSem (SemTrigger);
```

ENMClearSem - Clear a Semaphore

This call clears an OS/2 or AIX semaphore unconditionally. If processes are blocked on the semaphore, they are restarted.

C Definition

```
USHORT ENMClearSem (ULONG SemHandle);
```

Parameter Description

The following parameters are required:

- **retCode**(USHORT) - output

Code	Meaning
2	An internal error has occurred. The API program receives further information by calling the function ENMGetReason, described in “Handling Errors” on page 48. The reason code is also written to the diagnosis log on the S/390 (see “Chapter 11. Diagnosis Information” on page 71). If it is an internal error of the MERVA OS/2 V3 or MERVA AIX API, the reason code is zero.
Others	See the description of the OS/2 system call DosSemClear in the OS/2 online help or see the description of the ENMClearSem function in the book *MERVA AIX Application Programming*.

- **SemHandle**(ULONG) - input

  Generated by ENMCreateSem or ENMOpenSem.

C Language Example

```
/*
   If a connection to MERVA OS/2 V3 should be used,
   use the following define-statements:
*/
#define TRIGGER "\\SEM\\SAMPLE2"
/*
   If a connection to MERVA AIX should be used,
   use the following define-statements:
*/
/*
  #define TRIGGER "SAMPLE2"
*/
#include "enm4rapi.h"

   USHORT      rc = 0;
```

```
   ULONG       SemTrigger;

   if ((rc = ENMCreateSem (&SemTrigger, TRIGGER)) == 0)
      rc = ENMClearSem (SemTrigger);
```

ENMCreateSem - Create a Semaphore

This call creates an OS/2 or AIX semaphore. The semaphore is used by several API programs to synchronize their access to resources or to wait for MERVA OS/2 V3 or MERVA AIX alarms.

C Definition

```
USHORT ENMCreateSem (PULONG SemHandle,
                     PUCHAR SemName);
```

Parameter Description

The following parameters are required:

- **retCode**(USHORT) - output

  Code **Meaning**

  2 An internal error has occurred. The API program receives further information by calling the function ENMGetReason, described in “Handling Errors” on page 48. The reason code is also written to the diagnosis log on the S/390 (see “Chapter 11. Diagnosis Information” on page 71). If it is an internal error of the MERVA OS/2 V3 or MERVA AIX API, the reason code is zero.

  Others
  See the description of the OS/2 system call DosCreateSem in the OS/2 online help or see the description of the ENMCreateSem function in the book *MERVA AIX Application Programming*.

- **SemHandle**(PULONG) - output

  Address of the semaphore handle.

- **SemName**(PUCHAR) - input

  Pointer to a null-terminated string containing the name of the semaphore to be created.

  In MERVA AIX the semaphore name is a logical name without path details. In MERVA OS/2 V3 it must be a full path name and has to start with \SEM\.

C Language Example

```
/*
   If a connection to MERVA OS/2 V3 should be used,
   use the following define-statements:
*/
#define TRIGGER "\\SEM\\SAMPLE2"
/*
   If a connection to MERVA AIX should be used,
   use the following define-statements:
*/
/*
  #define TRIGGER "SAMPLE2"
*/
#include "enm4rapi.h"

   USHORT      rc = 0;
```

```
ULONG       SemTrigger;

rc = ENMCreateSem (&SemTrigger, TRIGGER);
```

ENMOpenSem - Open a Semaphore

This call opens an existing OS/2 or AIX semaphore created by another process with ENMCreateSem. The other process can also run on the OS/2 system or the RS/6000.

C Definition

```
USHORT ENMOpenSem (PULONG SemHandle,
                   PUCHAR SemName);
```

Parameter Description

The following parameters are required:

- **retCode**(USHORT) - output

Code	Meaning
2	An internal error has occurred. The API program receives further information by calling the function ENMGetReason, described in “Handling Errors” on page 48. The reason code is also written to the diagnosis log on the Enterprise System/9000 (see “Chapter 11. Diagnosis Information” on page 71). If it is an internal error of the MERVA OS/2 V3 or MERVA AIX API, the reason code is zero.
100	Limit of open semaphores in the system is exceeded.
123	The name for the semaphore is not valid.
187	The semaphore does not exist.
Others	See the description of the OS/2 system call DosOpenSem in the OS/2 online help or see the description of the ENMOpenSem function in the book *MERVA AIX Application Programming*.

- **SemHandle**(PULONG) - output

  Address of the handle of the opened semaphore.

- **SemName**(PUCHAR) - input

  Pointer to a null-terminated string containing the name of the semaphore to be opened.

C Language Example

```
/*
   If a connection to MERVA OS/2 V3 should be used,
   use the following define-statements:
*/
#define TRIGGER "\\SEM\\SAMPLE2"
/*
   If a connection to MERVA AIX should be used,
   use the following define-statements:
*/
/*
  #define TRIGGER "SAMPLE2"
*/
#include "enm4rapi.h"
```

```
USHORT      rc = 0;
ULONG       SemTrigger;

rc = ENMOpenSem (&SemTrigger, TRIGGER);
```

Handling Errors

If you want that the API call returns reason codes use the function ENMGetReason - Get Reason Code for Internal Error. This function is described in the following.

ENMGetReason - Get Reason Code for Internal Error

This call returns the reason code for an internal error in MERVA Connection/ESA.

If an internal error occurs either in MERVA Connection/ESA or in the local MERVA API, an API call returns the return code 2. If it is an error of the MERVA Connection/ESA, ENMGetReason returns a more specific reason code. Otherwise, the reason code is 0.

C Definition

```
USHORT ENMGetReason (void);
```

Parameter Description

The following parameter is required:

retCode(USHORT) - output

Code	Meaning
2xxx	All reason codes between 2000 and 2999 indicate communication problems.
2110	The APPC conversation cannot be established or is canceled.
2120	The Communications Side Information object is not found.
2130	Connection to Remote MERVA API Server program failed.
2140	Deallocation failed because the conversation has already been terminated.
2150	Conversation is interrupted while trying to receive data.
2200	An empty data buffer was received.
29xx	xx is a return code of the CPI-C call.
2999	A general communication problem occurred (see diagnosis log).
7006	The Remote MERVA API Server failed while allocating memory.
7012	The Remote MERVA API Server does not accept further API calls due to a previous error.
7013	The Remote MERVA API Server received a negative return code from user exit ENMExitDecrypt.
7014	The Remote MERVA API Server received a negative return code from user exit ENMExitEncrypt.
7015	The Remote MERVA API Server received a negative return code from user exit ENMExitMacVerify or ENMExitMacGen.

7016 The Remote MERVA API Server received an incorrect API request.

7018 The Remote MERVA API Server received an error when converting ASCII to EBCDIC. See the diagnosis log of MERVA OS/2 V3 or MERVA AIX.

7019 The Remote MERVA API Server received an error while accessing the message integrity control file.

7030 Internal message space has not been created.

8002 The Remote MERVA API Client cannot open the programmer's log file specified in the profile.

8003 The Remote MERVA API Client cannot close the programmer's log file specified in the profile.

8004 The Remote MERVA API Client cannot open the diagnosis log file specified in the profile.

8005 The Remote MERVA API Client cannot close the diagnosis log file specified in the profile.

8006 The Remote MERVA API Client could not allocate memory.

8007 The Remote MERVA API Client cannot write to the diagnosis log file specified in the profile.

8008 The Remote MERVA API Client cannot write to the programmer's log file specified in the profile.

8010 The Remote MERVA API Client failed because the profile name in ENMSetProfile was incorrect or was not specified.

8011 The Remote MERVA API Client failed because the profile specified in ENMSetProfile does not exist.

8013 The Remote MERVA API Client received a negative return code from user exit ENMExitDecrypt.

8014 The Remote MERVA API Client received a negative return code from user exit ENMExitEncrypt.

8015 The Remote MERVA API Client received a negative return code from user exit ENMExitMacVerify.

8016 The Remote MERVA API Client received a negative return code from user exit ENMExitMacGen.

8017 Conversation has not been started with ENMStartRAPI.

8019 The Remote MERVA API Client could not access the message integrity control file.

81xx The Remote MERVA API Client could not open the MERVA Connection/ESA profile. xx is the code contained in the CICS/ESA Exec Interface Block field EIBRESP after an EXEC CICS SET TDQUEUE request.

82xx The Remote MERVA API Client could not read the MERVA Connection/ESA profile. xx is the code contained in the CICS/ESA Exec Interface Block field EIBRESP after an EXEC CICS READQ TD request.

8300 The Remote MERVA API Client could not obtain data from storage which is to be written to the diagnosis log.

83xx The Remote MERVA API Client could not write data to the diagnosis log. xx is the code contained in the CICS/ESA Exec Interface Block field EIBRESP after an EXEC CICS WRITEQ TD request.

8400 The Remote MERVA API Client could not obtain data from storage which is to be written to the programmer's log.

84xx The Remote MERVA API Client could not write data to the programmer's log. xx is the code contained in the CICS/ESA Exec Interface Block field EIBRESP after an EXEC CICS WRITEQ TD request.

85xx The Remote MERVA API Client could not delete a record in the message integrity control file. xx is the code contained in the CICS/ESA Exec Interface Block field EIBRESP after an EXEC CICS DELETE DATASET request.

86xx The Remote MERVA API Client could not read a record from the message integrity control file. xx is the code contained in the CICS/ESA Exec Interface Block field EIBRESP after an EXEC CICS READ DATASET request.

87xx The Remote MERVA API Client could not write a record to the message integrity control file. xx is the code contained in the CICS/ESA Exec Interface Block field EIBRESP after an EXEC CICS WRITE DATASET request.

C Language Example

```
#include "enm4rapi.h"

   USHORT   rc     = 0;
   USHORT   reason = 0;

   rc = ENMFree();
   if (rc) {
      reason = ENMGetReason();
      if (reason) {
      printf ("Internal error in MERVA Connection/ESA occurred, reason code %d",
            reason);
      }
   }
```

Chapter 7. Resynchronization

If a network connection is interrupted, the recovery procedure must ensure that all changes of message status in MERVA OS/2 V3 or MERVA AIX (such as Add, Route, or Delete) are made only once. This affects both programs using the remote API and programs calling the local MERVA OS/2 V3 or MERVA AIX API.

During normal processing, an API call is transferred from the API socket to the Remote MERVA API Server (positions (1) and (2) in Figure 17). The return data from MERVA OS/2 V3 or MERVA AIX is transferred back from the Remote MERVA API Server to the Remote MERVA API Client (positions (3) and (4)) and to the calling program.

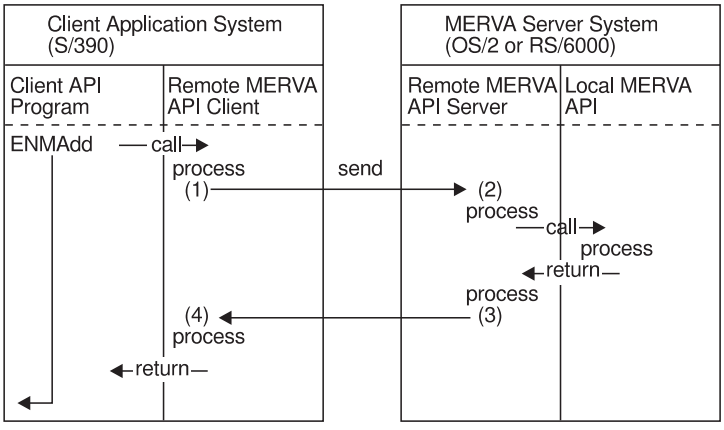

Figure 17. Resynchronization Support

The return code ERR_SYSTEM of the API call and a corresponding reason code (2000 to 2999) of an additional ENMGetReason call indicates whether the network connection is interrupted (see *MERVA OS/2 V3 Application Programming* or *MERVA AIX Application Programming*). MERVA Connection/ESA does not know whether the call completed successfully, unsuccessfully, or whether it is not executed on MERVA OS/2 V3 or MERVA AIX. In the example shown in Figure 17 this means that the API program does not know whether the message has been added to the MERVA OS/2 V3 or MERVA AIX queue.

With MERVA Connection/ESA the API program reestablishes the connection in the next run using ENMRestartRAPI. It recreates the message with the same contents and fields, and repeats the call that failed. This mechanism is provided for those API calls that are important for the integrity of the message database:

- ENMAdd
- ENMDelete
- ENMPut
- ENMRouteAdd
- ENMRoutePut

How Resynchronization Is Implemented

The Remote MERVA API Client generates an internal unique identifier when it receives a call from the application program. The identifier is saved locally and also sent to the Remote MERVA API Server. The Remote MERVA API Server deletes the identifier after the API call has been executed and the return data is passed back to the Remote MERVA API Client.

If the network connection terminates before the return data is passed back, identifier and return data are saved. After the connection is reestablished, the same identifier arrives with the first of the above mentioned API calls. The saved return data is passed back as if the call was executed now.

The necessary control data is saved in files. On the Remote MERVA API Client you can specify the file name in the MERVA Connection/ESA profile as described in “Settings in the MERVA Connection/ESA Profile” on page 29. On the Remote MERVA API Server the file name must be the same as the application name specified in the ENMStartRAPI or ENMRestartRAPI call.

To ensure that resynchronization works correctly, note the following:

- The Message Integrity Control file (MIP) **must not be a recoverable resource** (in CICS terms).

  The MIP file contains all information required for a resynchronization. When a transaction fails, changes to recoverable resources are backed out by the CICS dynamic transaction backout. Thus the saved resynchronization information might be lost.

  This also applies to other resources used in the API transaction for a proper restart of the transaction after a program or system failure.

If you run more than one remote API programs,

- You can share the same MIP file between the API programs.

  This requires that each API program has been assigned a different transaction code (or that a single API program has been assigned several transaction codes). The transaction code is used as a key for a control record in the MIP file.

  The MIP file is a VSAM key-sequenced data set (KSDS). The key has a length of 8 bytes. The first 4 bytes are made up of the transaction code. The last 4 bytes contain the terminal identifier if the transaction was started by entering the transaction code at the terminal. When the transaction was started another way the last 4 bytes of the key consist of four blank characters.
- You must use unique application names for the ENMStartRAPI and ENMRestartRAPI calls.

  As well as for the MIP file key, you can use the transaction code as a unique application name. You obtain the transaction code from the CICS/ESA Exec Interface Block field EIBTRNID. Look into the source code of sample program ENMVERIF on page 87 where this method is used.

Using the Resynchronization Mechanism

The following is a program that issues calls in a loop:

```
ENMSetProfile
ENMRestartRAPI
```

```
ENMAttach
do
   ENMCreate
   ENMWriteField
   read message from application
   ENMRouteAdd
until (no more message to send)
ENMDetach
ENMEndRAPI
```

If the network connection breaks down after the ENMRouteAdd call is issued, the API program terminates. When it is restarted, the loop is entered as if there had been no interruption in the previous run.

Notes:

1. Use the same profile as in the previous run.
2. Call ENMRestartRAPI using the same application name.
3. Call ENMCreate and ENMWriteField using the same data as in the previous run (same message, same field contents).
4. Call ENMRouteADD using the same queue name.
5. After resynchronization continue with the loop as in normal processing.

If the program runs like that, it does not have to check how far processing went in the previous run when the ENMRouteADD call was interrupted.

Hints and Tips

Recovering after a Failed Call

If calling ENMAdd or ENMRouteAdd fails, you usually call ENMClear to clear the message space (see *MERVA OS/2 V3 Application Programming* or *MERVA AIX Application Programming*).

If these calls fail after reestablishing the connection as described before because of other reasons than network problems, calling ENMClear may return the return code

`ERR_NO_MSG_CREATED`

(that is, 202). This means that the API call was executed in the first run. The error message can be ignored.

The same applies to an ENMFree call returning the message

`ERR_NO_MSG_LOCKED`

(that is, 201) after calling of ENMDelete, ENMPut, or ENMRoutePut failed.

Not Using Resynchronization

If you do not use the resynchronization option, call ENMStartRAPI instead of ENMRestartRAPI. ENMStartRAPI deletes the internal control information for resynchronization. Then each API call is considered as a new one.

MERVA Connection/ESA does not save the type or input data of the API call that failed due to the network failure. Therefore, when using ENMRestartRAPI, you must ensure that the same call is repeated after reconnecting to MERVA OS/2 V3 or MERVA AIX if one of the above mentioned calls failed.

MERVA Connection/ESA does not recognize an inappropriate API call. The call is not executed if the internal state indicates that the last API call from the previous run was executed. If this is not considered, an API call with new data could be treated as a repeated call from a previous run.

Chapter 8. Security

Security is a fundamental requirement for all financial institutions. When discussing the security of message transfers, a number of different aspects must be considered:

- Encryption of transferred information
- Authentication of transferred information.

These requirements are supported by MERVA Connection/ESA.

Encryption of Transferred Information

Using MERVA Connection/ESA you can encrypt all information that is exchanged.

You do this by activating user exits. User exits allow you to include your own algorithm or even other products that support encryption and decryption routines.

There are two user exits:

- ENM4ExitEncrypt for encryption
- ENM4ExitDecrypt for decryption.

See “User Exit Interfaces” for more information on how to implement these routines.

Authentication of Transferred Information

Using MERVA Connection/ESA you can generate an authentication key covering all exchanged information. You do this by activating user exits. User exits allow you to include your own algorithm or even other products that support authentication routines.

There are two user exits:

- ENM4ExitMacGen for MAC generation
- ENM4ExitMacVerify for MAC verification.

See “User Exit Interfaces” for more detailed information on how to implement these routines.

User Exit Interfaces

The following introduces the user exit interfaces of MERVA Connection/ESA.

Introduction

There is a fundamental difference between an API call and a user exit:

- For an API call, you write a program that calls the API routine provided by MERVA Connection/ESA.
- A user exit is a routine written by you and called by MERVA Connection/ESA. The user exit routines must contain the declaration for the function name and formal parameter list, as described in the following.

User Exit Points

Figure 18 shows what happens when an API function is called by an API program on the S/390. You can see who is calling a user exit at which processing step. In the figure, the following abbreviations are used for the user exits:

ENCRYP ENM4ExitEncrypt

DECRYP ENM4ExitDecrypt

MACGEN ENM4ExitMacGen

MACVFY ENM4ExitMacVerify

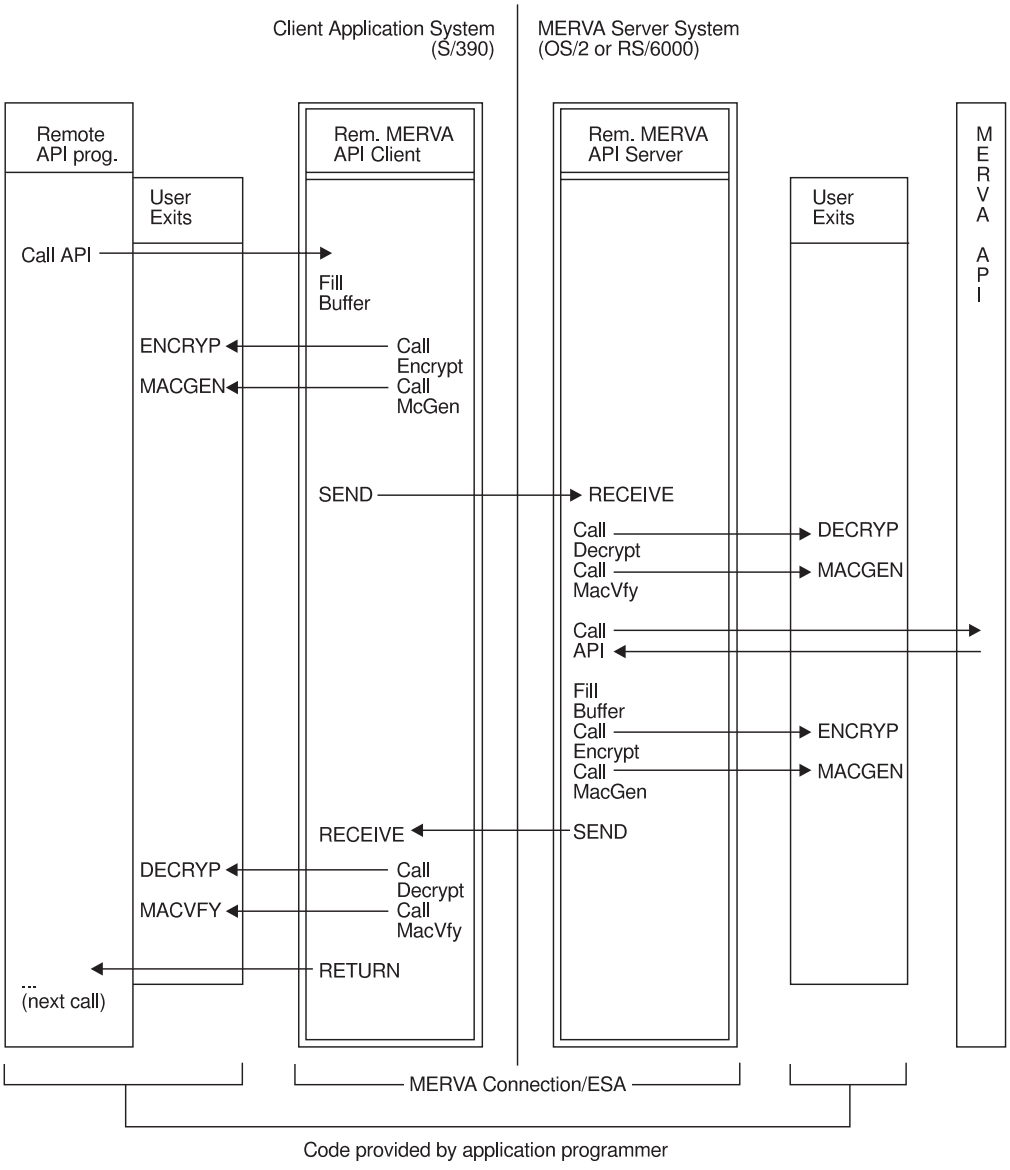

Figure 18. User Exit Points

User Exit Interfaces in C Language

The data types used in these routines can be different, depending on whether they are implemented on the OS/2 system, the RS/6000 or the S/390. See the coded samples (“Appendix C. Sample Security User Exits” on page 83) for more information.

User Exit for Encryption

C Definition

```
unsigned short ENM4ExitEncrypt ( unsigned char* pucApplId,
                                 unsigned char* pucBuffer,
                                 unsigned short usBufferLen );
```

Purpose of the User Exit Routine

Encrypts the passed data buffer.

Parameter Description

The following parameters are required:

- **pucApplId**(unsigned char*) - input

  Address of a null-terminated string with a maximum length of 8. The string contains the application identifier that you passed as a parameter of the API call ENMStartRAPI. You can use this string to provide different encryption keys for different partner connections, or decide for which connections or for which API programs the information is to be encrypted.
- **pucBuffer**(unsigned char*) - input/output

  Address of the data buffer to be encrypted.
- **usBufferLen**(unsigned short) - input

  Length of the data buffer to be encrypted.

User Exit for Decryption

C Definition

```
unsigned short ENM4ExitDecrypt ( unsigned char* pucApplId,
                                 unsigned char* pucBuffer,
                                 unsigned short usBufferLen );
```

Purpose of the User Exit Routine

Decrypts the passed data buffer.

Parameter Description

The following parameters are required:

- **pucApplId**(unsigned char*) - input

  Address of a null-terminated string with a maximum length of 8. The string contains the application identifier that you passed as a parameter of the API call ENMStartRAPI. You can use this string to provide different decryption keys for different partner connections, or to decide for which connections or for which API programs the information is to be decrypted.
- **pucBuffer**(unsigned char*) - input, output

  Address of the data buffer to be decrypted.

- **usBufferLen**(unsigned short) - input

  Length of the data buffer to be decrypted.

User Exit for MAC Generation

C Definition

```
unsigned short ENM4ExitMacGen  ( unsigned char* pucApplId,
                                 unsigned char* pucBuffer,
                                 unsigned short usBufferLen,
                                 unsigned char* pucMacBuffer);
```

Purpose of the User Exit Routine

Generates a MAC (Message Authentication Code) for the passed data buffer.

Parameter Description

The following parameters are required:

- **pucApplId**(unsigned char*) - input

  Address of a null-terminated string with a maximum length of 8. The string contains the application identifier you passed as a parameter of the API call ENMStartRAPI. You can use this string to provide different MAC generation algorithms for different partner connections, or to decide for which connections or for which API programs a MAC shall be generated.
- **pucBuffer**(unsigned char*) - input

  Address of the data buffer for which to generate a MAC.
- **usBufferLen**(unsigned short) - input

  Length of the data buffer for which to generate a MAC.
- **pucMacBuffer**(unsigned char*) - output

  Address of the area to copy the generated MAC to. The address can be up to 32 bytes in length.

User Exit for MAC Verification

C Definition

```
unsigned short ENM4ExitMacVerify  (unsigned char* pucApplId,
                                   unsigned char* pucBuffer,
                                   unsigned short usBufferLen,
                                   unsigned char  pucMacBuffer);
```

Purpose of the User Exit Routine

Generates a MAC for the passed data buffer and compares it with the passed MAC. Set the return code to 0 if the MAC matches and otherwise to 1.

Parameter Description

The following parameters are required:

- **pucApplId**(unsigned char*) - input

  Address of a null-terminated string with a maximum length of 8. The string contains the application identifier you passed as a parameter of the API call ENMStartRAPI. You can use this string to provide different MAC verification algorithms for different partner connections, or to decide on which connections or for which API programs a MAC is to be verified.

- **pucBuffer**(unsigned char*) - input

  Address of the data buffer for which to generate a MAC and for which the passed MAC has been generated on the partner side.
- **usBufferLen**(unsigned short) - input

  Length of the data buffer for which to generate a MAC.
- **pucMacBuffer**(unsigned char*) - input

  Address of the area holding the MAC key that has been received from the partner. The address can be up to 32 bytes in length.

ENM4ExitMacVerify

Chapter 9. Building API Programs

This chapter describes how to compile MERVA Connection/ESA API programs in the C/370 programming language. The programs may contain EXEC CICS statements.

Compiling the Sample Programs

The following list shows the sample programs delivered with MERVA Connection/ESA. You find the programs in the library with the low level qualifier SENMSRC0. The programs which use EXEC CICS statements are indicated.

List of Sample Programs

ENMVERIF Sample program to verify correct installation. It contains EXEC CICS statements.

ENMABEND Sample program (abnormal termination exit) to verify correct installation. It contains EXEC CICS statements.

ENMSAMP Sample program to send telex messages to MERVA

ENM4SSEC Sample security user exit routines

ENM4SNIL Skeleton for security user exit routines.

The first three programs are available in your installation as executable modules. ENM4SSEC is link-edited to ENMSAMP, and ENM4SNIL is link-edited to ENMVERIF and to ENMABEND. So you need not compile any of these programs. However, if you want to modify one or more of the security user exit routines, you have to compile and link-edit them.

Compiling a MERVA Connection/ESA API Program

The following JCL shows, how you can compile and link-edit an API program which does not contain EXEC CICS statements. The IBM-supplied cataloged procedures EDCC (Compile) and EDCPL (Prelink and Link-edit) are used. You find this job as member ENMCCOMP in the library with the low level qualifier SENMINS0.

```
//ENMCC    JOB (ACCT),NAME,MSGCLASS=L,CLASS=Z                   MODIFY
//*********************************************************************
//*     COMPILE AND LINK-EDIT A CONNECTION/ESA API PROGRAM
//*     ------------------------------------------------------
//*     PLEASE REMOVE COMMENTS FROM CONTROL CARDS BEFORE JOB START
//*     ------------------------------------------------------
//*********************************************************************
//CC       EXEC EDCC,
//         CPARM='RENT,SO,OPT,DEF(CICS),MAR(1,80),NOSEQ',      ( 1)
//         OUTFILE='Your Object Library(Member),DISP=OLD',     ( 2)
//         SYSOUT1=A,    OUTPUT CLASS FOR MESSAGES
//         SYSOUT6=A     OUTPUT CLASS FOR LISTING
//COMPILE.SYSLIB  DD
//                DD DSN=CICS330.SDFHC370,DISP=SHR             ( 3)
//COMPILE.USERLIB DD DSN=Your User Include File,DISP=SHR       ( 4)
//COMPILE.SYSIN   DD DATA,DLM=$$
                  .
      Your C/370 Source Code
                  .
$$
//*--------------------------------------------------------------------
//*    PRELINK AND LINK-EDIT THE API PROGRAM                          -
//*--------------------------------------------------------------------
//PL       EXEC EDCPL,COND.PLKED=(4,LE,CC.COMPILE),
//         COND.LKED=((4,LE,CC.COMPILE),(4,LT,PLKED)),
//         LPARM='LIST,MAP,LET,XREF,RENT,AMODE=31,RMODE=ANY',  ( 5)
//         OUTFILE='Your Load Library(Member),DISP=OLD'        ( 6)
//*
//*    PRELINK THE API PROGRAM
//*
//PLKED.SYSLIB DD DSN=Your Object Library,DISP=SHR             ( 7)
//PLKED.SYSIN  DD *
  INCLUDE SYSLIB(Member)                                     ( 8)
  INCLUDE SYSLIB(ENMRAPI,ENMRUTL,ENMRPRF,ENMSSEC)            ( 9)
/*
//*
//*    LINK-EDIT THE API PROGRAM
//*
//LKED.SYSLIB DD
//            DD
//            DD DSN=CICS330.SDFHLOAD,DISP=SHR                (10)
//LKED.SYSLIN DD
//            DD DDNAME=SYSIN
//LKED.SYSIN  DD *
  INCLUDE SYSLIB(DFHELII,DFHCPLC)                            (11)
  ORDER DFHELII,CEESTART
  ENTRY CEESTART
/*
//
```

Figure 19. Compile API Program without EXEC CICS Statements

Comments:

(1) The compiler options RENT, DEF(CICS), MAR(1,80), and NOSEQ are required.

(2) Specify the name of the file where the object module is to be stored, and the name of the object module.

(3) This is the CPI-C pseudonym file. For CICS/ESA Version 3 Release 2.1 the high level qualifier is CICS321. Normally, you do not require the pseudonym file. It is for your information only.

(4) Specify the name of the file which contains the MERVA Connection/ESA include-files and your own include-files.

(5) The linkage editor options RENT, AMODE=31, and RMODE=ANY are required.

(6) Specify the name of the file where the load module is to be stored, and the name of the load module.

(7) Specify the name of the file where the object module was stored in the compile step (CC.COMPILE).

(8) Specify the name of the object module.

(9) The first three MERVA Connection/ESA modules must always be included. If you need no security exits, include ENMSNIL instead of ENMSSEC.

(10) This is the CICS load library. For CICS/ESA Version 3 Release 2.1 the high level qualifier is CICS321.

(11) You must always include the EXEC interface stub DFHELII and the CICS CPI Communications stub DFHCPLC. DFHELII must be placed before CEESTART.

The following JCL shows, how you can compile and link-edit an API program with embedded EXEC CICS statements. The IBM-supplied cataloged procedure DFHEITDL is used. You find this job as member ENMCCICS in the library with the low level qualifier SENMINS0.

```
//ENMCTRN  JOB (ACCT),NAME,MSGCLASS=L,CLASS=Z                   MODIFY
//*****************************************************************
//*  TRANSLATE, COMPILE, LINK-EDIT A CONNECTION/ESA API PROGRAM
//*  -----------------------------------------------------------
//*  PLEASE REMOVE COMMENTS FROM CONTROL CARDS BEFORE JOB START
//*  -----------------------------------------------------------
//*****************************************************************
//CCTRAN  EXEC DFHEITDL,
//         CVER='C/370 Version',                                 ( 1)
//         PARM.TRN='MAR(1,80,0),NSEQ,OM(1,80,0),NOS',           ( 2)
//         CPARM='RENT,SO,OPT,DEF(CICS),MAR(1,80),NOSEQ',        ( 3)
//         LNKPARM='LIST,MAP,LET,XREF,RENT,AMODE=31,RMODE=ANY'   ( 4)
//***    LPARM='LIST,MAP,LET,XREF,RENT,AMODE(31),RMODE(ANY)''    ( 5)
//*
//*--------------------------------------------------------------------
//*    TRANSLATE AND COMPILE THE API PROGRAM                          -
//*--------------------------------------------------------------------
//*
//*    TRANSLATE THE API PROGRAM
//*
//TRN.SYSIN  DD DATA,DLM=$$
             .
      Your C/370 Source Code
             .
$$
//*
//*    COMPILE THE API PROGRAM
//*
//C.USERLIB    DD DSN=Your User Include File,DISP=SHR            ( 6)
//C.SYSLIN     DD DSN=Your Object Library(Member),DISP=OLD       ( 7)
//*
//*--------------------------------------------------------------------
//*    PRELINK and LINK-EDIT THE API PROGRAM                          -
//*--------------------------------------------------------------------
//*
//*    PRELINK THE API PROGRAM
//*
//PLKED.SYSLIB DD DSN=Your Object Library,DISP=SHR               ( 8)
//PLKED.SYSIN  DD *
  INCLUDE SYSLIB(Member)                                       ( 9)
  INCLUDE SYSLIB(ENMRAPI,ENMRUTL,ENMRPRF,ENMSSEC)              (10)
/*
//*
//*    LINK-EDIT THE API PROGRAM
//*
//LKED.SYSLMOD DD DSN=Your Load Library,DISP=OLD                (11)
//LKED.SYSIN DD *
  INCLUDE SYSLIB(DFHELII,DFHCPLC)                              (12)
  ORDER DFHELII,CEESTART
  ENTRY CEESTART
  NAME Member(R)                                               (13)
/*
//
```

Figure 20. Compile API Program Containing EXEC CICS Statements

Comments:

(1) Specify the appropriate version of the C/370 compiler according to the conventions in your installation.

(2) All specified CICS translator options are required.

(3) The compiler options RENT, DEF(CICS), MAR(1,80), and NOSEQ are required.

(4) The linkage editor options RENT, AMODE=31, and RMODE=ANY are required.

(5) This is the parameter for the linkage editor options when you compile the program for CICS/ESA Version 3, Release 2.1. Again, the linkage editor options RENT, AMODE(31), and RMODE(ANY) are required.

(6) Specify the name of the file which contains the MERVA Connection/ESA include-files and your own include-files.

(7) Specify the name of the file where the object module is to be stored, and the name of the object module.

(8) Specify the name of the file where the object module was stored in the compile step (CCTRAN.C).

(9) Specify the name of the object module.

(10) The first three MERVA Connection/ESA modules must always be included. If you need no security exits, include ENMSNIL instead of ENMSSEC.

(11) Specify the name of the file where the load module is to be stored.

(12) You must always include the EXEC interface stub DFHELII and the CICS CPI Communications stub DFHCPLC. DFHELII must be placed before CEESTART.

(13) Specify the name of the load module.

Chapter 10. Replacing Security User Exits

This chapter describes how you can replace the provided security user exits by generating and activating your own security user exits on the S/390, the OS/2 system, or the RS/6000.

Security User Exits

Two sets of sample security user exits are provided (see “User Exit Interfaces” on page 55):

enm4ssec These routines contain sample code for encryption and authentication. They show how to access the variables of the formal parameter list in the function call but do not provide genuine security. The provided code on the S/390 is named ENMSSEC. It is contained in the object library with the low level qualifier SENMOBJ0. On the OS/2 system it is the dynamic link library enm4ssec.dll. On the RS/6000 it is the shared library libenmssec.a.

enm4snil These routines do not contain any code. Use this file if no encryption or authentication is desired. The provided code on the S/390 is named ENMSNIL. It is also contained in the object library with the low level qualifier SENMOBJ0. On the OS/2 system it is the dynamic link library enm4snil.dll. On the RS/6000 it is the shared library libenmsnil.a.

On the OS/2 system the dynamic link library containing the user exits must have the name enm4sxit.dll. The shipped version of enm4sxit.dll is a copy of the sample library enm4snil.dll. If you want to use the second set (enm4ssec) of sample user exit routines, copy enm4ssec.dll to enm4sxit.dll.

On the RS/6000 the shared library containing the user exits must have the name libenmsxit.a. The shipped version of libenmsxit.a is a copy of the sample library libenmsnil.a. If you want to use the second set (libenmssec) of sample user exit routines, copy libenmssec.a to libenmsxit.a.

Generating and Activating Security User Exits on the S/390

On the S/390 the user exit routines must be placed in an object library.

To replace the sample user exits by your own routines, use ENM4SSEC from the library with the low level qualifier SENMSRC0 as a skeleton. Insert your code in ENM4SSEC and compile the module using procedure EDCC. Then you can prelink and link-edit ENM4SSEC to your API program using procedure EDCPL (provided that the API program is already contained in an object library). You find examples for the required JCL in “Compiling a MERVA Connection/ESA API Program” on page 61.

Generating and Activating Security User Exits on the OS/2 System

On the OS/2 system the user exit routines must be placed in DLLs.

If you want to replace the sample user exits with your own routines, use enm4ssec.c as a skeleton. The following file generates a new enm4ssec.dll from enm4ssec.c:

```
#------------------------------------------------------------------------
#  ENM4SSEC.MAK  - Make file to generate a DLL with Security User Exits
#------------------------------------------------------------------------
#------------------------------------------------------------------------
#  Compile-Options:
#  /C+     Compile, do not link
#  /Gd-    Static linking of the runtime library
#  /Sp1    Structure alignment set to 1-byte boundaries to be compatible
#          with the 16-bit code ( /ZP option ) of MERVA OS/2 V3
#  /Se     Allow all C Set/2 language extensions except migration
#  /Ss+    Allow use of double slashes (//) for comments
#  /Re     Generate executable code for C Set/2 runtime environment
#  /Gm+    Use the multithread version of the libraries
#  /Kb+    Produce basic diagnostic messages
#  /Fo+    Create an object file
#  /Ti+    Generate debugger information
#  /Ge-    Build a .DLL file
#
#  /DOS2   enable 'define INCL_BASE', 'include <OS2.H>' - see ENM4SSEC.C
#
#  Link-Options:
#  /NOE[EXTDICTIONARY]       Ignore Extended Dictionary
#  /NOD[EFAULTLIBRARYSEARCH] Ignore Default Libraries
#  /NOI[GNORECASE]           Case sensitive
#  /CO[DEVIEW]               Include symbolic debugging information
#  /BATCH                    Return error, if input file name is missing
#  /A[LIGNMENT]              Alignment Factor in the executable, 512 is default
#  /E[XEPACK]                Packing Executable Files
#  /STACK                    Stack size
#------------------------------------------------------------------------

C_OPT= /C+ /Gd- /Sp1 /Se /Ss+ /Re /Gm+ /Kb+ /Fo+ /Ti+ /Ge- /DOS2
L_OPT= /NOE /NOD /NOI /CO /BATCH /A:512 /E /STACK:16384

#------------------------------------------------------------------------
#  Objects which are to be generated in this make file
#------------------------------------------------------------------------
ALL:  ENM4SSEC.DLL ENM4SXIT.LIB

#------------------------------------------------------------------------
#  Link ENM4SSEC.DLL
#------------------------------------------------------------------------
ENM4SSEC.DLL: ENM4SSEC.OBJ ENM4SXIT.DEF
              -LINK386 $(L_OPT) ENM4SSEC.OBJ,ENM4SSEC.DLL,ENM4SSEC.MAP,\
                   OS2386.LIB + DDE4MBS.LIB,ENM4SXIT.DEF >>ENM4SSEC.LOG;2>&1

#------------------------------------------------------------------------
#  Compile ENM4SSEC
#------------------------------------------------------------------------
ENM4SSEC.OBJ: ENM4SSEC.C  ENM4SXIT.H
              icc $(C_OPT) ENM4SSEC.C >>ENM4SSEC.LOG;2>&1

#------------------------------------------------------------------------
#  Generate LIB file for exit DLL
#------------------------------------------------------------------------
ENM4SXIT.LIB: ENM4SXIT.DEF
              IMPLIB ENM4SXIT.LIB ENM4SXIT.DEF
```

Figure 21. Make File to Generate a DLL

Use the following command to create the new file:

make /f enm4ssec.mak

If there are error messages, they are written to the file enm4ssec.log. Copy the newly generated enm4ssec.dll to enm4sxit.dll.

If your source file name is different from enm4ssec.c, replace every occurrence of enm4ssec within the make file enm4ssec.mak with your program name.

Generating and Activating Security User Exits on the RS/6000

The sample security user exits can be accessed by the Remote MERVA API Server on the RS/6000 if you copy the library libenmssec.a to the library libenmsxit.a. If you want to replace the sample user exits by your own routines, use the enm4ssec.c as a skeleton. The file can be retrieved from directory */usr/lpp/merva/samples*. The file enm4ssec.mak generates a new library libenmssec.a from the source file enm4ssec.c.

Use the following command: **make -f enm4ssec.mak all**

Replace /usr/lpp/merva/lib/libenmsxit.a with your new library using the following command:

cp libenmssec.a /usr/lpp/merva/lib/libenmsxit.a

Chapter 11. Diagnosis Information

This chapter describes the diagnosis information that is written to log files on the S/390, the OS/2 system, or the RS/6000.

Log Files on the S/390

Two logs are written. In the MERVA Connection/ESA profile, you can:

- Set their names
- Set a logging level between 1 and 4.

The profile contents is described in “Settings in the MERVA Connection/ESA Profile” on page 29.

Diagnosis Log

The diagnosis log provides you with:

- Error messages that help you recover from errors when using the API calls or errors concerning the communication with the OS/2 system, or the RS/6000.
- Trace information when the API Trace is switched on with the call ENMTrace (see *MERVA OS/2 V3 Application Programming* or *MERVA AIX Application Programming*).

Programmer’s Log

The programmer’s log is a general debugging tool. It contains all entries of the diagnosis log and additional more detailed information to be analyzed by your IBM representative.

By the logging level in the MERVA Connection/ESA profile, you can influence how much log information MERVA Connection/ESA is to provide:

- Level 1 should be used in general. No data is written to the log.
- Level 2 and 3 are reserved for future use. No data is written to the log.
- Level 4 should be used when you want to record the activities of MERVA Connection/ESA in a comprehensive way. This level is suited for debugging or demonstration purposes (for example, for the MERVA Connection/ESA installation verification).

Log Message Layout

Each message written to the logs consists of two parts, the message header and the message body, as shown in Figure 22 on page 72.

```
*  TXCD  19970402192358ENM4RAPI ENMRestartRAPI 00000 00000
ENM9153: API function ENMRestartRAPI called.
         Parameters:
         App: SAMPLE3

*  TXCD  19970402192358ENM4RUTL APIInit 00000 00000
ENM9108: Error in CPIC Call  CMALLC  RC = 19.

*  TXCD  19970402192413ENM4RAPI ENMRestartRAPI
ENM9109: Error in APPC Initialization.

*  TXCD  19970402192413ENM4RAPI ENMRestartRAPI
ENM9152: API function returned with reason code 2130.
```

Figure 22. Example of Diagnosis Log with API Trace Entries

The layout of the message header is as follows:

***** Start of header.

Transaction code
The CICS transaction code is a 1- to 4-character code identifying the API program.

Terminal ID The CICS terminal identifier is a 4-character code identifying the terminal where the API program is run.

This item is **optional**. It is part of the message header only when the CICS task was associated with a terminal. In Figure 22 you see that there is a transaction code in the message header, but no terminal ID.

Date The date is in the form YYYYMMDD, where YYYY is the year, MM is the month, and DD is the day.

Time The time is in the form HHMMSS, where HH is the hour, MM are the minutes, and SS are the seconds.

Module name The module name is an 8-character code identifying the module the message originated from.

Function name
The function name is a 15-character code identifying the function the message originated from.

The layout of the message body is as follows:

Message The variable-length message to be recorded. See the *MERVA AIX Application Programming*, *MERVA OS/2 V3 Messages and Codes* or the *MERVA AIX Messages and Codes* manual for the meaning of the messages.

Note: Log entries are appended to the existing files when you specify parameter `DISP=MOD` in the appropriate DD statements for both log files in your CICS startup job. When you use `DISP=OLD` or `DISP=SHR`, MERVA Connection/ESA starts from the beginning of the file and overwrites previous with new log entries.

Log Files on MERVA OS/2 V3

Diagnosis information concerning the Remote MERVA API Server program is provided by the MERVA OS/2 V3 log files. Error and trace information is written to the diagnosis log. IBM service information is written to the programmer's log. You can list or browse the diagnosis log file using the Display/Print Diagnosis Log (DPD) function of MERVA OS/2 V3.

The log files are located on the disk and directory *x*:\MERVA2\, where *x* is the drive on which MERVA OS/2 V3 is installed. See the *MERVA OS/2 V3 Diagnosis Guide* for further information.

Log Files on MERVA AIX

Diagnosis information concerning the Remote MERVA API Server program is provided by the MERVA AIX log files. Error and trace information is written to the diagnosis log. IBM service information is written to the programmer's trace log. You can list or browse the diagnosis log file using the Display Diagnosis Log function in the MERVA AIX menu program.

The log files are located in the MERVA AIX instance logging directory as selected in the *Create MERVA AIX Instance* step described in the *MERVA AIX Installation and Customization Guide*.

Appendix A. Sample Network Definitions for the S/390

This appendix contains sample network definitions for the S/390. First you see VTAM definitions, then CICS definitions follow.

Note: Please keep in mind that the following definitions are examples only. They are provided to support you when you establish an APPC connection between the S/390 and the OS/2 system. Before you try this, consult the VTAM and CICS system specialist to get the required information how to adapt the provided examples.

VTAM Definitions: OS/2 System Connected to a Token Ring Network

This example contains the VTAM definition of a CICS application, a PU macro defined for a Token Ring, and the logmode entries of an APPC connection from CICS to an OS/2 system which is part of a Token Ring network.

You find these definitions as members ENMAPPL, ENMPULU, and ENMLOGMO in the library with the low level qualifier SENMMAC0.

CICS Application Definition

```
*=====================================================================*
* - CICS-Application                                                  *
*=====================================================================*
   VBUILD TYPE=APPL
*                                                                     *
I40AC388 APPL  AUTH=(PASS,ACQ,VPACE),VPACING=5,    CICS APPLID         C
               ACBNAME=I40AC388,                   CICS APPLID         C
               DLOGMOD=CICSISC,                    LOGMODE NAME        C
               MODETAB=PPC3270,PARSESS=YES
```

PU/LU Definition for the OS/2 System in a Token Ring

```
*=====================================================================*
* - PU-Definitions for the OS/2 system                                *
* - LU-Definitions for the OS/2 system                                *
*=====================================================================*
*                                                                     *
         VBUILD TYPE=SWNET,                                          C
               MAXNO=12,                                             C
               MAXGRP=5
*                                                                     *
FD5773   PU    ADDR=01,                          PHYSICAL UNIT (PU)   C
               ANS=CONT,                                             C
               IDBLK=05D,                                            C
               IDNUM=FE573,                      LOCAL NODE ID       C
               ISTATUS=ACTIVE,                                       C
               DLOGMOD=MOD2,                                         C
               MAXDATA=1024,                                         C
               MAXOUT=1,                                             C
               MODETAB=MTGADL,                                       C
               USSTAB=USSSNA,                                        C
               PUTYPE=2
*
FD577300 LU    LOCADDR=0,DLOGMOD=CICSISC     LU NAME OF OS/2 MACHINE
*                                            AND LOGMODE NAME
FD577302 LU    LOCADDR=2
*                                                                     *
```

Logmode Entry for an APPC Connection

```
*====================================================================*
* - Logmodes for an APPC connection to an S/390                *
* - Both Logmodes CICSISC and SNASVCMG are necessary                 *
*====================================================================*
PPC3270  MODETAB
*                                                                    *
CICSISC  MODEENT LOGMODE=CICSISC,                      LOGMODE NAME    C
               COS=INTERACT,FMPROF=X'13',TSPROF=X'07',                 C
               PRIPROT=X'B0',SECPROT=X'B0',COMPROT=X'D0B1',            C
               PSERVIC=X'060200000000000000000300',RUSIZES=X'8585',    C
               PSNDPAC=X'04',SRCVPAC=X'04',SSNDPAC=X'01',TYPE=X'00'
SNASVCMG MODEENT LOGMODE=SNASVCMG,FMPROF=X'13',TSPROF=X'07',           C
               PRIPROT=X'B0',SECPROT=X'B0',COMPROT=X'D0B1',            C
               SSNDPAC=X'01',                                          C
               SRCVPAC=X'04',                                          C
               PSNDPAC=X'04',                                          C
               RUSIZES=X'8585',PSERVIC=X'060200000000000000000300',    C
               ENCR=B'0000'
*====================================================================*
END      MODEEND
         END   PPC3270
```

VTAM Definitions: OS/2 System Connected by an SDLC Line

The CICS application definition and the logmode entries are the same as for an OS/2 system connected to a Token Ring network. The PU macro, however, is different.

CICS Application Definition

```
*====================================================================*
* - CICS-Application                                                 *
*====================================================================*
   VBUILD TYPE=APPL
*                                                                    *
I40AC388 APPL  AUTH=(PASS,ACQ,VPACE),VPACING=5,    CICS APPLID         C
               ACBNAME=I40AC388,                   CICS APPLID         C
               DLOGMOD=CICSISC,                    LOGMODE NAME        C
               MODETAB=PPC3270,PARSESS=YES
```

PU/LU Definition for the OS/2 System with SDLC Line

```
*====================================================================*
* - PU-Definitions for the OS/2 system with SDLC line                *
* - LU-Definitions for the OS/2 system with SDLC line                *
*====================================================================*
                  .........
F38L0036 LINE  ADDRESS=(036,HALF)                  SDLC LINE 036
         SERVICE ORDER=(FD5773)                    PHYSICAL UNIT (PU)
FD5773   PU    PUTYPE=2,ADDR=C1,                   PHYSICAL UNIT (PU) C
               DLOGMOD=MOD2,MAXDATA=521,                              C
               MAXOUT=7,PASSLIM=7,SSCPFM=USSSCS,ISTATUS=ACTIVE,       C
               MODETAB=MTGADL,XID=YES
FD577300 LU    LOCADDR=0,ISTATUS=ACTIVE,DLOGMOD=CICSISC  LU NAME OF OS/2
*                                          MACHINE AND LOGMODE NAME
FD577302 LU    LOCADDR=2,ISTATUS=ACTIVE
*
```

Logmode Entry for an APPC Connection

```
*====================================================================*
* - Logmodes for an APPC connection to an OS/2 system                *
* - Both Logmodes CICSISC and SNASVCMG are necessary                 *
```

```
*====================================================================*
PPC3270  MODETAB
*                                                                    *
CICSISC  MODEENT LOGMODE=CICSISC,                      LOGMODE NAME    C
               COS=INTERACT,FMPROF=X'13',TSPROF=X'07',                 C
               PRIPROT=X'B0',SECPROT=X'B0',COMPROT=X'D0B1',            C
               PSERVIC=X'060200000000000000000300',RUSIZES=X'8585',    C
               PSNDPAC=X'04',SRCVPAC=X'04',SSNDPAC=X'01',TYPE=X'00'
SNASVCMG MODEENT LOGMODE=SNASVCMG,FMPROF=X'13',TSPROF=X'07',           C
               PRIPROT=X'B0',SECPROT=X'B0',COMPROT=X'D0B1',            C
               SSNDPAC=X'01',                                          C
               SRCVPAC=X'04',                                          C
               PSNDPAC=X'04',                                          C
               RUSIZES=X'8585',PSERVIC=X'060200000000000000000300',    C
               ENCR=B'0000'
*====================================================================*
END      MODEEND
         END   PPC3270
```

CICS Definitions

With the following job you can define an APPC connection from CICS/ESA to a remote OS/2 system, or RS/6000. The DEFINE statements are part of the member ENMCSD contained in the library with the low level qualifier SENMMAC0.

In this sample four parallel sessions will be defined (see parameter MAXIMUM(4,2)). For the logmode parameter MODENAME(CICSISC) and the OS/2 system machine LU name parameter NETNAME(FD577300), you can find the corresponding entries in “VTAM Definitions: OS/2 System Connected to a Token Ring Network” on page 75 or in “VTAM Definitions: OS/2 System Connected by an SDLC Line” on page 76.

```
//UPDCSD   EXEC PGM=DFHCSDUP,REGION=4096K
//STEPLIB  DD   DSN=CICS330.SDFHLOAD,DISP=SHR     OR CICS321
//SYSPRINT DD   SYSOUT=*
//DFHCSD   DD   DSN=CICSxxx.CSD,DISP=SHR
//SYSIN    DD   *
*
DELETE GROUP(ENMGROUP)
*
* CONNECTION/ESA COMMUN. DEFINITIONS FOR MERVA OS/2 V3 and MERVA AIX
*
DEFINE PROFILE(ENMPROF)  GROUP(ENMGROUP)
       MODENAME(CICSISC)                      <= LOGMODE NAME
*
DEFINE CONNECTION(RAPI)  GROUP(ENMGROUP)
       NETNAME(FD577300)                      <= OS/2 or RS/6000 LU NAME
       ACCESSMETHOD(VTAM)
       PROTOCOL(APPC)
       SINGLESESS(NO)
       DATASTREAM(USER)
       RECORDFORMAT(U)
       AUTOCONNECT(ALL)
       INSERVICE(YES)
       ATTACHSEC(LOCAL)
       BINDSECURITY(NO)
*
DEFINE SESSIONS(C3RAPI)  GROUP(ENMGROUP)
       CONNECTION(RAPI)                       <= SEE PREVIOUS DEFINE
       MODENAME(CICSISC)                      <= LOGMODE NAME
       PROTOCOL(APPC)
       MAXIMUM(4,2)
       SENDSIZE(256)
       RECEIVESIZE(256)
```

```
       SESSPRIORITY(0)
       AUTOCONNECT(ALL)
       BUILDCHAIN(YES)
       USERAREALEN(0)
       IOAREALEN(0,0)
       RELREQ(NO)
       DISCREQ(NO)
       NEPCLASS(0)
       RECOVOPTION(SYSDEFAULT)
       RECOVNOTIFY(NONE)
*
DEFINE PARTNER(M2API)    GROUP(ENMGROUP)      <= COMM. SIDE INFORMATION
       NETNAME(FD577300)                      <= OS/2 or RS/6000 LU NAME
       PROFILE(ENMPROF)                       <= SEE PREVIOUS DEFINE
       TPNAME(MERVA2)                         <= TP NAME OF API SERVER (MERVA OS/2 V3)
*      TPNAME(ENMRAS)                         <= TP NAME OF API SERVER (MERVA AIX)
*
ADD    GROUP(ENMGROUP) LIST(XXXLIST)
/*
//
```

Appendix B. Sample Network Definitions for the RS/6000

This appendix contains sample network definitions for the RS/6000.

Note: Please keep in mind that the following definitions are examples only. They are provided to support you when you establish an APPC connection between the S/390 and the RS/6000.

Control Point Definition

```
control_pt:
        prof_name                                       = "node_cp"
        xid_node_id                                     = 0x071fea1d
        network_name                                    = "DEIBMFD"
        control_pt_name_alias                           = "FDA71D"
        control_pt_name                                 = "FDA71D"
        control_pt_node_type                            = appn_end_node
        max_cached_trees                                = 500
        max_nodes_in_topology_database                  = 500
        route_addition_resistance                       = 128
        comments                                        = "CP for ESA Connection over SNA"
```

Local LU Definition

```
local_lu_lu6.2:
        prof_name                                       = "MERESALLUP"
        local_lu_name                                   = "FDA71D00"
        local_lu_alias                                  = "FDA71D00"
        local_lu_dependent                              = no
        local_lu_address                                =
        sscp_id                                         = *
        link_station_prof_name                          = ""
        conversation_security_list_profile_name         = ""
        rrm_enabled                                     = no
        comments                                        = "RemAPI with CICS/ESA system"
```

Partner LU Definition

```
partner_lu6.2:
        prof_name                                       = "ESAPLUP"
        fq_partner_lu_name                              = "DEIBMID.I40AC388"
        partner_lu_alias                                = "I40AC388"
        session_security_supp                           = no
        parallel_session_supp                           = yes
        conversation_security_level                     = already_verified
        comments                                        = "RemAPI with CICS/ESA system"
```

Transaction Program Definition

```
local_tp:
        prof_name                                       = "ENMRAS"
        tp_name                                         = "ENMRAS"
        tp_name_in_hex                                  = no
        pip_data_present                                = no
        pip_data_subfields_number                       = 0
        command_line_parameters_present                 = yes
        command_line_parameters                         = "trace"
        conversation_type                               = mapped
        sync_level                                      = confirm
        resource_security_level                         = none
        resource_access_list_profile_name               = ""
        full_path_tp_exe                                = "/home/merva1/ipc/enmtpi.cmd"
```

```
        multiple_instances                      = yes
        user_id                                 = 210
        server_synonym_name                     = "ENMRASRV"
        restart_action                          = once
        communication_type                      = signals
        ipc_queue_key                           = 0
        attach_timeout                          = yes
        attach_timeout_value                    = 60
        standard_input_device                   = "/dev/null"
        standard_output_device                  = "/dev/null"
        standard_error_device                   = "/dev/null"
        comments                                = "RemAPI server program"
```

Token Ring Link Station Definition

```
link_station_token_ring:
        prof_name                               = "tokesa"
        use_control_pt_xid                      = yes
        xid_node_id                             = "*"
        sna_dlc_profile_name                    = "tok0"
        stop_on_inactivity                      = no
        time_out_value                          = 0
        LU_registration_supported               = no
        LU_registration_profile_name            = ""
        link_tracing                            = no
        trace_format                            = long
        access_routing_type                     = link_address
        remote_link_name                        = ""
        remote_link_address                     = 0x400010000008
        remote_sap                              = 0x04
        call_out_on_activation                  = yes
        verify_adjacent_node                    = no
        net_id_of_adjacent_node                 = ""
        cp_name_of_adjacent_node                = ""
        xid_node_id_of_adjacent_node            = "*"
        node_type_of_adjacent_node              = learn
        solicit_sscp_sessions                   = yes
        activate_link_during_system_init        = no
        activate_link_on_demand                 = no
        cp_cp_sessions_supported                = yes
        cp_cp_session_support_required          = no
        adjacent_node_is_preferred_server       = no
        initial_tg_number                       = 0
        restart_on_normal_deactivation          = no
        restart_on_abnormal_deactivation        = no
        restart_on_activation                   = no
        TG_effective_capacity                   = 4300800
        TG_connect_cost_per_time                = 0
        TG_cost_per_byte                        = 0
        TG_security                             = nonsecure
        TG_propagation_delay                    = lan
        TG_user_defined_1                       = 128
        TG_user_defined_2                       = 128
        TG_user_defined_3                       = 128
        comments                                = "Link Station for CICS/ESA"
```

Token Ring SNA DLC Definition

```
sna_dlc_token_ring:
        prof_name                               = "tok0"
        datalink_device_name                    = "tok0"
        force_timeout                           = 120
        user_defined_max_i_field                = no
        max_i_field_length                      = 30729
        max_active_link_stations                = 100
        num_reserved_inbound_activation         = 0
```

```
num_reserved_outbound_activation              = 0
transmit_window_count                         = 127
dynamic_window_increment                      = 1
retransmit_count                              = 8
receive_window_count                          = 1
priority                                      = 0
inact_timeout                                 = 48
response_timeout                              = 4
acknowledgement_timeout                       = 1
link_name                                     = ""
local_sap                                     = 0x04
retry_interval                                = 60
retry_limit                                   = 20
dynamic_link_station_supported                = yes
trace_base_listen_link_station                = no
trace_base_listen_link_station_format         = long
dynamic_lnk_solicit_sscp_sessions             = yes
dynamic_lnk_cp_cp_sessions_supported          = yes
dynamic_lnk_cp_cp_session_support_required    = no
dynamic_lnk_TG_effective_capacity             = 4300800
dynamic_lnk_TG_connect_cost_per_time          = 0
dynamic_lnk_TG_cost_per_byte                  = 0
dynamic_lnk_TG_security                       = nonsecure
dynamic_lnk_TG_propagation_delay              = lan
dynamic_lnk_TG_user_defined_1                 = 128
dynamic_lnk_TG_user_defined_2                 = 128
dynamic_lnk_TG_user_defined_3                 = 128
comments                                      = ""
```

Mode Definition

```
mode:
        prof_name                             = "CICSISCP"
        mode_name                             = "CICSISC"
        max_sessions                          = 8
        min_conwinner_sessions                = 4
        min_conloser_sessions                 = 4
        auto_activate_limit                   = 8
        max_adaptive_receive_pacing_window    = 16
        receive_pacing_window                 = 7
        max_ru_size                           = 30720
        min_ru_size                           = 256
        class_of_service_name                 = "#CONNECT"
        comments                              = "RemAPI with CICS/ESA system"
```

Appendix C. Sample Security User Exits

This appendix contains listings of sample security user exits that you can use. You find the security user exits in the library with the low level qualifier SENMSRC0.

Module ENM4SNIL - Empty Functions

This program is integrated into MERVA Connection/ESA in the supplied version. No actions are taken in the functions. This means that data transferred between the S/390 and the OS/2 system, or RS/6000 is not encrypted and no authentication key is built or transferred. You can use this program as a skeleton for your code.

```
/*--------------------------------------------------------------------*\
| ENM4SNIL                                                            |
\*--------------------------------------------------------------------*/
#if defined(OS2)
#define  INCL_BASE
#include <OS2.H>
#endif

#include "enm4sxit.h"

#ifndef __32BIT__
   #define APIENTRY16 APIENTRY
   #define PUCHAR16   PUCHAR
#endif

USHORT APIENTRY16 ENM4ExitMacGen  (PUCHAR16 pucApplId,
                                   PUCHAR16 pucBuffer,
                                   USHORT   usBufferLen,
                                   PUCHAR16 pucMacBuffer)
{
   return(0);
}

USHORT APIENTRY16 ENM4ExitMacVerify  (PUCHAR16 pucApplId,
                                      PUCHAR16 pucBuffer,
                                      USHORT   usBufferLen,
                                      PUCHAR16 pucMacBuffer)
{
   return(0);
}

USHORT APIENTRY16 ENM4ExitEncrypt ( PUCHAR16 pucApplId,
                                    PUCHAR16 pucBuffer,
                                    USHORT   usBufferLen )
{
   return(0);
}

USHORT APIENTRY16 ENM4ExitDecrypt ( PUCHAR16 pucApplId,
                                    PUCHAR16 pucBuffer,
                                    USHORT   usBufferLen )
{
   return(0);
}
```

Figure 23. Sample Security User Exit ENM4SNIL

Module ENM4SSEC - Sample Functions

This module is supplied as an example for coding security functions. Simple encryption and authentication routines are included. However, they do not provide genuine security.

```
/*--------------------------------------------------------------------*\
| ENM4SSEC                                                           |
\*--------------------------------------------------------------------*/
#if defined(OS2)
#define  INCL_BASE
#include <OS2.H>
#endif
#include<string.h>
#include "enm4sxit.h"
/* defines that this module can be compiled with Cset/2 and IBM C/2 */
#ifndef __32BIT__
   #define APIENTRY16 APIENTRY
   #define PUCHAR16   PUCHAR
#endif
#if defined(CICS)
#pragma map(Enm36Table,"Enm36Tab")
#endif
unsigned char Enm36Table[36]= {'\x00', '\x01', '\x02', '\x03',
                               '\x04', '\x05', '\x06', '\x07',
                               '\x08', '\x09', '\x0A', '\x0B',
                               '\x0C', '\x1D', '\x1E', '\x1F',
                               '\x10', '\x11', '\x12', '\x13',
                               '\x14', '\x15', '\x16', '\x17',
                               '\x18', '\x19', '\x1A', '\x1B',
                               '\x1C', '\x1D', '\x1E', '\x1F',
                               '\x20', '\x21', '\x22', '\x23' };
#define ENM_MAX_BASE              36
#define ENM_FILL_CHAR             0

#if defined(CICS)
#pragma map(EnmBasestr,"ENMBASTR")
#endif
unsigned short EnmBasestr(unsigned short base,
                          unsigned long  num,
                          unsigned char* basestring,
                          unsigned short max_len)
{
   unsigned long  count=0,reminder=0;
   short          position;
   unsigned long  number;
   number = num;
   position = max_len-1;
   memset (basestring, ENM_FILL_CHAR, max_len);
   basestring[position]=0;

   if (base > ENM_MAX_BASE) return(1);
   do {
      if (--position < 0) return(1);
      reminder = number % (unsigned long)base;
      count = number / (unsigned long)base;
      if (!count) {
          basestring[position++]=Enm36Table[reminder];
          break;
      }
      basestring[position]=Enm36Table[reminder];
      number = count;
   } while (1);
   return(0);
}
```

Figure 24. Sample Security User Exit ENM4SSEC (Part 1 of 2)

```
USHORT APIENTRY16 ENM4ExitMacGen  (PUCHAR16 pucApplId,
                                   PUCHAR16 pucBuffer,
                                   USHORT   usBufferLen,
                                   PUCHAR16 pucMacBuffer){
   register       i;
   unsigned long  ulAddedByteValues=0;
   unsigned short rc = 0;

   if (!strcmp(pucApplId,"APPLAUTH") || !strcmp(pucApplId,"APPLSECR")) {
      for (i=0;i<usBufferLen;i++) {
         ulAddedByteValues += (unsigned long) pucBuffer[i];
      }
      rc = EnmBasestr(2,
                      ulAddedByteValues,
                      pucMacBuffer,
                      32);
   }
   return(rc);
}

USHORT APIENTRY16 ENM4ExitMacVerify  (PUCHAR16 pucApplId,
                                      PUCHAR16 pucBuffer,
                                      USHORT   usBufferLen,
                                      PUCHAR16 pucMacBuffer)
{
   register       i;
   unsigned long  ulAddedByteValues=0;
   unsigned char  ucaCalcMacBuffer[32];
   unsigned short rc = 0;

   if (!strcmp(pucApplId,"APPLAUTH") || !strcmp(pucApplId,"APPLSECR")) {
      for (i=0;i<usBufferLen;i++) {
         ulAddedByteValues += (unsigned long) pucBuffer[i];
      }
      memset (ucaCalcMacBuffer,0,32);

      rc = EnmBasestr(2,
                      ulAddedByteValues,
                      ucaCalcMacBuffer,
                      32);
      if (!rc) rc = memcmp(ucaCalcMacBuffer,pucMacBuffer,32);
   }
   return(rc);
}
USHORT APIENTRY16 ENM4ExitEncrypt ( PUCHAR16 pucApplId,
                                    PUCHAR16 pucBuffer,
                                    USHORT   usBufferLen )
{
   register i;

   if (!strcmp(pucApplId,"APPLENCR") || !strcmp(pucApplId,"APPLSECR")) {
      for (i=0;i<usBufferLen;i++) {
         pucBuffer[i] = pucBuffer[i] ¬ 255; /* negation */
      }
   }
   return(0);
```

Figure 24. Sample Security User Exit ENM4SSEC (Part 2 of 2)

Appendix D. Sample Programs

This appendix includes listings of sample MERVA Connection/ESA API programs written in C/370. You find the sample programs in the library with the low level qualifier SENMSRC0.

Program ENMVERIF

```
/*****************************************************************************/
/*                                                                           */
/* PROGRAM NAME: ENMVERIF                                                    */
/* -------------                                                             */
/*  Installation verification program for Connection/ESA.                    */
/*                                                                           */
/* COPYRIGHT:                                                                */
/* ----------                                                                */
/*  (C) Copyright International Business Machines Corporation 1994, 1997     */
/*                                                                           */
/* REVISION LEVEL: 3.0                                                       */
/* ---------------                                                           */
/*                                                                           */
/* WHAT THIS PROGRAM DOES:                                                   */
/* -----------------------                                                   */
/*  This program verifies whether the connection between an application      */
/*  running under CICS/ESA and the API of MERVA was set up properly.         */
/*                                                                           */
/*  It requires that a MERVA user "SAMPLE" with password "SAMPLE1"           */
/*  exists. User "SAMPLE" must be approved to start an API application       */
/*  program.                                                                 */
/*  In addition, the queue API_IN must have been customized.                 */
/*                                                                           */
/* FILE MEMBERS NEEDED TO COMPILE:                                           */
/* -------------------------------                                           */
/*    ENMVERIF       - This file                                             */
/*    ENM4RAPI       - The API include file                                  */
/*    The preprocessor of CICS/ESA V3.2.1 or later and                       */
/*    the AD/Cycle C/370 compiler are required.                              */
/*                                                                           */
/* PROGRAMS TO BE LINKED:                                                    */
/* ----------------------                                                    */
/*                                                                           */
/*    ENMRAPI                                                                */
/*    ENMRPRF                                                                */
/*    ENMRUTL                                                                */
/*    ENMSNIL                                                                */
/*    In addition, the stubs DFHELII and DFHCPLC are required.               */
/*                                                                           */
/*                                                                           */
```

Figure 25. Sample Program ENMVERIF (Part 1 of 4)

```
/* REQUIRED INPUT:                                                          */
/* ---------------                                                          */
/*  Enter the transaction code ENM2 at the CICS terminal.                   */
/*                                                                          */
/* EXPECTED OUTPUT:                                                         */
/* ----------------                                                         */
/*  The confirmation that the connection to the MERVA API                   */
/*  could be established and released, and that the queue API_IN exists.    */
/*  The appropriate information is displayed on the terminal.               */
/*  Alternatively, the information is printed at the transient data         */
/*  destination COUT (or whichever destination represents the               */
/*  C/370 stdout).                                                          */
/*  The last line to be displayed or printed must read:                     */
/*      "Transaction ENM2 has ended".                                       */
/*                                                                          */
/* MERVA CALLS USED:                                                        */
/* -----------------                                                        */
/*    ENMAttach      - Attach to MERVA                                      */
/*    ENMDetach      - Detach from MERVA                                    */
/*    ENMQueryQueue  - Query queue information                              */
/*                                                                          */
/* ADDITIONAL Connection/ESA CALLS USED:                                    */
/* -------------------------------------                                    */
/*    ENMSetProfile  - Select the profile to be used                        */
/*    ENMStartRAPI   - Establish connection to MERVA                        */
/*    ENMEndRAPI     - Disconnect from MERVA                                */
/*    ENMGetReason   - Get reason code for internal error                   */
/*                                                                          */
/****************************************************************************/

#include <stdio.h>
#include <string.h>
#include "enm4rapi.h"

#define LENGTH 81

/* Function to write a message line to the screen or to print it      */
void SendText(UCHAR Message??(??));

main()
{
   USHORT rc = 0;                    /* Return code of API calls     */
   SHORT  rs = 0;                    /* Reason code of API calls     */
   USHORT Messagecount;              /* Number of messages in a queue*/

   UCHAR  ProfName??(5??) = "PROF";  /* Connection profile name      */
   UCHAR  Message??(LENGTH??);       /* Message line on screen       */
```

Figure 25. Sample Program ENMVERIF (Part 2 of 4)

```
EXEC CICS ADDRESS EIB(dfheiptr);   /* Get address of the CICS EIB  */

sprintf(Message,"Transaction %s has started...",dfheiptr->eibtrnid);
SendText(Message);                 /* Send first line to screen    */

/* Specify the name of the Connection/ESA profile                  */
ENMSetProfile (ProfName);          /* Extrapartition TD queue name */

sprintf(Message,"Profile name %s specified",ProfName);
SendText(Message);
/* Establish connection to MERVA                                   */
if ((rc = ENMStartRAPI (dfheiptr->eibtrnid)) == 0) {
   sprintf(Message,"APPC Conversation named %s is up",dfheiptr->eibtrnid);
   SendText(Message);

   /* Attach to MERVA                                              */
   if ((rc = ENMAttach ("SAMPLE","SAMPLE1","API")) == 0) {
      strcpy(Message,"Program attached to MERVA");
      SendText(Message);

      /* Check existence of queue API_IN                           */
      if ((rc = ENMQueryQueue ("API_IN",&Messagecount)) == 0)
         strcpy(Message,"Queue API_IN exists");
      else
         if ((rs = ENMGetReason ()) == 0)
            sprintf(Message,
                    "MERVA API: Error in ENMQueryQueue, rc = %d",rc);
         else
            sprintf(Message,
                    "Error in ENMQueryQueue, rc = %d, rs = %d",rc,rs);
      SendText(Message);

      /* Detach from MERVA                                         */
      if ((rc = ENMDetach ()) == 0)
         strcpy(Message,"Program detached from MERVA");
      else
         if ((rs = ENMGetReason ()) == 0)
            sprintf(Message,
                    "MERVA API: Error in ENMDetach, rc = %d",rc);
         else
            sprintf(Message,
                    "Error in ENMDetach, rc = %d, rs = %d",rc,rs);
   } else
      if ((rs = ENMGetReason ()) == 0)
         sprintf(Message,"MERVA API: Error in ENMAttach, rc = %d",rc);
      else
         sprintf(Message,"Error in ENMAttach, rc = %d, rs = %d",rc,rs);
   SendText(Message);
```

Figure 25. Sample Program ENMVERIF (Part 3 of 4)

```
      /* Disconnect from MERVA                                        */
      if ((rc = ENMEndRAPI ()) == 0)
         strcpy(Message,"APPC Conversation successfully terminated");
      else
         if ((rs = ENMGetReason ()) == 0)
            sprintf(Message,"MERVA API: Error in ENMEndRAPI, rc = %d",rc);
         else
            sprintf(Message,"Error in ENMEndRAPI, rc = %d, rs = %d",rc,rs);
   } else
      if ((rs = ENMGetReason ()) == 0)
         sprintf(Message,"MERVA API: Error in ENMStartRAPI, rc = %d",rc);
      else
         sprintf(Message,"Error in ENMStartRAPI, rc = %d, rs = %d",rc,rs);
   SendText(Message);

   sprintf(Message,"Transaction %s has ended",dfheiptr->eibtrnid);
   SendText(Message);                /* Send last line to screen     */
}
/*********************************************************************/
/* Send a message line to the terminal.                              */
/* If an exceptional condition occurs the message line is printed.   */
/*********************************************************************/
void SendText(UCHAR Message??(??))
{
   static UCHAR  NLMsg??(LENGTH+23??);/* Work field for message line  */
   static USHORT Line = 0;           /* Line number on screen        */
          ULONG  resp, resp2;        /*RESP and RESP2 after CICS call*/

   EXEC CICS HANDLE ABEND            /* Activate abnormal term. exit */
        PROGRAM("ENMABEND")
        RESP(resp)
        RESP2(resp2);

   if (resp == DFHRESP(NORMAL))
      if (++Line == 1)               /* Erase screen when first line */
         EXEC CICS SEND TEXT
              FROM(Message)
              LENGTH(strlen(Message))
              ERASE
              RESP(resp)
              RESP2(resp2);
      else {                         /* Follow-on line               */
         NLMsg??(Line-2??) = '\n';
         NLMsg??(Line-1??) = '\0';
         strcat(NLMsg,Message);      /* '\n' occurs (Line-1) times   */

         EXEC CICS SEND TEXT
              FROM(NLMsg)
              LENGTH(strlen(NLMsg))
              WAIT
              RESP(resp)
              RESP2(resp2);
      }

   if (resp != DFHRESP(NORMAL)) {    /* Displayed successfully ?     */
      printf("%s\n",Message);        /* No, print the message        */
      EXEC CICS SEND CONTROL ALARM;  /* Indicate it by audible alarm */
   }

   EXEC CICS HANDLE ABEND            /* Cancel abnormal term. exit   */
        CANCEL;
}
```

Figure 25. Sample Program ENMVERIF (Part 4 of 4)

Program ENMABEND

```
/*****************************************************************************/
/* PROGRAM NAME: ENMABEND                                                    */
/* -------------                                                             */
/*  Abnormal termination exit for the Connection/ESA                         */
/*  installation verification program ENMVERIF.                              */
/*                                                                           */
/* COPYRIGHT:                                                                */
/* ----------                                                                */
/*  (C) Copyright International Business Machines Corporation 1994, 1997     */
/*                                                                           */
/* REVISION LEVEL: 3.0                                                       */
/* ---------------                                                           */
/*                                                                           */
/* WHAT THIS PROGRAM DOES:                                                   */
/* -----------------------                                                   */
/*  This program prints the results of the Connection/ESA                    */
/*  installation verification at the transient data destination COUT         */
/*  (or whichever destination represents the C/370 stdout).                  */
/*  It requires that a MERVA user "SAMPLE" with password "SAMPLE1"           */
/*  exists. User "SAMPLE" must be approved to start an API application       */
/*  program.                                                                 */
/*  In addition, the queue API_IN must have been customized.                 */
/*                                                                           */
/*  ENMABEND gets control only after abnormal termination of the             */
/*  installation verification program ENMVERIF. A possible reason may be that */
/*  the minimum level of the CICS Basic Mapping Support (BMS=MINIMUM)        */
/*  was specified at the CICS system generation or in the SIT (System        */
/*  Initialization Table).                                                   */
/*                                                                           */
/* FILE MEMBERS NEEDED TO COMPILE:                                           */
/* -------------------------------                                           */
/*    ENMABEND       - This file                                            */
/*    ENM4RAPI       - The API include file                                 */
/*    The preprocessor of CICS/ESA V3.2.1 or later and                      */
/*    the AD/Cycle C/370 compiler are required.                              */
/*                                                                           */
/* PROGRAMS TO BE LINKED:                                                    */
/* ----------------------                                                    */
/*                                                                           */
/*    ENMRAPI                                                                */
/*    ENMRPRF                                                                */
/*    ENMRUTL                                                                */
/*    ENMSNIL                                                                */
/*    In addition, the stubs DFHELII and DFHCPLC are required.               */
/*                                                                           */
/* REQUIRED INPUT:                                                           */
/* ---------------                                                           */
/*  None.                                                                    */
/*                                                                           */
/* EXPECTED OUTPUT:                                                          */
/* ----------------                                                          */
/*  The confirmation that the connection to the MERVA API                    */
/*  could be established and released, and that the queue API_IN exists.     */
/*  The last line to be printed must read:                                   */
/*      "Transaction ENM2 has ended".                                        */
/*                                                                           */
```

Figure 26. Sample Program ENMABEND (Part 1 of 3)

```
/* MERVA CALLS USED:                                                          */
/* -----------------                                                          */
/*    ENMAttach      - Attach to MERVA                                        */
/*    ENMDetach      - Detach from MERVA                                      */
/*    ENMQueryQueue  - Query queue information                                */
/*                                                                            */
/* ADDITIONAL Connection/ESA CALLS USED:                                      */
/* -------------------------------------                                      */
/*    ENMSetProfile  - Select the profile to be used                          */
/*    ENMStartRAPI   - Establish connection to MERVA                          */
/*    ENMEndRAPI     - Disconnect from MERVA                                  */
/*    ENMGetReason   - Get reason code for internal error                     */
/****************************************************************************/
#include <stdio.h>
#include "enm4rapi.h"

main()
{
   USHORT rc = 0;                    /* Return code of API calls     */
   SHORT  rs = 0;                    /* Reason code of API calls     */
   USHORT Messagecount;              /* Number of messages in a queue*/

   UCHAR  ProfName??(5??) = "PROF";  /* Connection profile name      */

   EXEC CICS ADDRESS EIB(dfheiptr);  /* Get address of the CICS EIB  */

   printf("Transaction %s has started...\n",dfheiptr->eibtrnid);

   /* Specify the name of the Connection/ESA profile                 */
   ENMSetProfile (ProfName);         /* Extrapartition TD queue name */

   printf("Profile name %s specified\n",ProfName);
```

Figure 26. Sample Program ENMABEND (Part 2 of 3)

```
   /* Establish connection to MERVA                                    */
   if ((rc = ENMStartRAPI (dfheiptr->eibtrnid)) == 0) {
      printf("APPC Conversation named %s is up\n",dfheiptr->eibtrnid);

      /* Attach to MERVA                                               */
      if ((rc = ENMAttach ("SAMPLE","SAMPLE1","API")) == 0) {
         printf("Program attached to MERVA\n");

         /* Check existence of queue API_IN                            */
         if ((rc = ENMQueryQueue ("API_IN",&Messagecount)) == 0)
            printf("Queue API_IN exists\n");
         else
            if ((rs = ENMGetReason ()) == 0)
               printf("MERVA API: Error in ENMQueryQueue, rc = %d\n",rc);
            else
               printf("Error in ENMQueryQueue, rc = %d, rs = %d\n",rc,rs);

         /* Detach from MERVA                                          */
         if ((rc = ENMDetach ()) == 0)
            printf("Program detached from MERVA\n");
         else
            if ((rs = ENMGetReason ()) == 0)
               printf("MERVA API: Error in ENMDetach, rc = %d\n",rc);
            else
               printf("Error in ENMDetach, rc = %d, rs = %d\n",rc,rs);
      } else
         if ((rs = ENMGetReason ()) == 0)
            printf("MERVA API: Error in ENMAttach, rc = %d\n",rc);
         else
            printf("Error in ENMAttach, rc = %d, rs = %d\n",rc,rs);

      /* Disconnect from MERVA                                         */
      if ((rc = ENMEndRAPI ()) == 0)
         printf("APPC Conversation successfully terminated\n");
      else
         if ((rs = ENMGetReason ()) == 0)
            printf("MERVA API: Error in ENMEndRAPI, rc = %d\n",rc);
         else
            printf("Error in ENMEndRAPI, rc = %d, rs = %d\n",rc,rs);
   } else
      if ((rs = ENMGetReason ()) == 0)
         printf("MERVA API: Error in ENMStartRAPI, rc = %d\n",rc);
      else
         printf("Error in ENMStartRAPI, rc = %d, rs = %d\n",rc,rs);

   printf("Transaction %s has ended\n",dfheiptr->eibtrnid);
}
```

Figure 26. Sample Program ENMABEND (Part 3 of 3)

Appendix E. 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
500 Columbus Avenue
Thornwood, NY 10594
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:

- ACF/VTAM
- AD/Cycle
- AIX
- AIX/6000
- AS/400
- AT
- C/2
- C/370
- C/400
- CICS
- CICS/ESA
- COBOL/400
- DATABASE 2
- DB2
- IBM
- MERVA
- MVS/ESA
- MVS/SP

- Operating System/2
- OS/2
- OS/400
- RACF
- RISC System/6000
- RPG/400
- RS/6000
- S/390
- SAA
- Series/1
- Systems Application Architecture
- 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.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Other company, product, and service names may be trademarks or service marks of others.

Glossary of Terms and Abbreviations

This glossary defines terms and abbreviations as they are used in this book. If you do not find the terms you are looking for, refer to *Dictionary of Computing*, New York: McGraw-Hill, 1994, the *S.W.I.F.T. User Handbook*, or the *S.W.I.F.T. USE Planning Guide*.

A

Advanced Program-to-Program Communication. A communications architecture that allows transaction programs to exchange information on a peer-to-peer basis. SNA LU 6.2 allows APPC architecture to operate on an SNA network.

Remote MERVA API Client. MERVA Application Programming Interface on the S/390.

Remote MERVA API Server. A program of MERVA Connection/ESA that is installed and runs on OS/2 or AIX. It communicates with the Remote MERVA API Client on the S/390 system.

API. application program interface.

application program interface. (1) A set of run-time routines or system calls that allows an application program to use a particular service provided by either the operating system or another licensed program. (2) The formally defined programming language interface that is between a system control program or a licensed program and the user of the program.

APPC. Advanced Program-to-Program Communications.

C

call. To activate a program or procedure, usually by specifying the entry conditions and jumping to an entry point.

Common Programming Interface. An interface providing languages and services that can be used to develop applications that take advantage of Systems Application Architecture (SAA) consistency.

Communications Side Information. An object in CPI Communications containing initialization parameters. These are, for example:

- The name of the partner program (for example, of the Remote MERVA API Server) with which a program can establish a conversation
- The name of the logical unit (LU) at the partner program's node, which CPI Communications requires to establish a conversation.

CPI. Common Programming Interface.

CPI Communications. Provides a consistent application programming interface for applications that require program-to-program communication. The interface makes use of the SNA LU 6.2 protocol to create a rich set of interprogram services.

CPI-C. CPI Communications.

CSI. Communications Side Information.

customization. The process of describing optional changes to defaults of a software program that is already installed on the system and configured so that it can be used.

customize. (1) To describe to the system the devices, programs, users, and user defaults for a particular data processing system or network. (2) To describe optional preferences or changes to defaults in a software program that is already installed and configured.

E

EPM. Extended Programming Model.

H

handle. A data structure that is a temporary local identifier for an object. You create a handle by allocating it. You make a handle identify an object at a specific location by binding it.

I

identifier. (1) A name you use to refer to a data object. An identifier contains some combination of letters, digits, and underscores, but its first character cannot be a digit. (2) In programming languages, a lexical unit that names a language object, such as the name of an array, record, label, or procedure. An identifier usually begins with a letter optionally followed by letters, digits, or other characters. (3) A sequence of bits or characters that identifies a program, device, or system to another program, device, or system.

include file. A text file that contains declarations used by a group of functions, programs, or users.

I

Input Sequence Number (ISN). A sequential number that identifies a message sent to the SWIFT network.

J

JCL. Job Control Language.

L

logical unit. (1) A type of network addressable unit that enables end users to communicate with each other and gain access to network resources. (2) In SNA, a port through which an end user accesses the SNA network in order to communicate with another user, and through which the end user accesses the functions provided by system services control points (SSCPs). A LU can support at least two sessions, one with an SSCP and one with another LU, and may be capable of supporting many sessions with other LUs.

loop. A sequence of instructions performed repeatedly until an ending condition is reached.

LU. logical unit.

M

MAC. Message Authentication Code.

Message Authentication Code. A code of a specific length that is calculated with a particular algorithm from a message buffer. It is sent with the message. The partner recalculates it and compares it with the received code. This allows modifications of the transferred data to be detected.

Message Reference Number. A unique 16-digit number assigned by MERVA to each message for identification purposes. The message reference number consists of an 8-digit domain identifier and an 8-digit sequential number.

MRN. Message Reference Number.

N

node. An end point of a link, or a junction common to two or more links in a network. Nodes can be processors, controllers, or workstations, and they can vary in routing and other functional capabilities.

P

partner. In data communications, the remote application program or the remote computer.

peer-to-peer communications. Pertaining to data communications between two nodes that have equal status in the interchange. Either node can begin the conversation.

S

semaphore. (1) Entity used to control access to system resources. Processes can be locked to a resource with semaphores if the processes follow certain programming conventions. (2) Provides a general method to synchronize two processes.

SNA. System Network Architecture.

System Network Architecture. (1) An architecture for controlling the transfer of information in a data communications network. (2) The description of the logical structure, formats, protocols, and operating sequences for transmitting information units through, and controlling the configuration and operation of, networks.

Bibliography

IBM Publications

With exception of the General Information and the Licensed Program Specifications all MERVA books are available as softcopy on the

- *MERVA Family C-Kit*, SK2T-0157

MERVA Family Books

- *MERVA OS/2 Client User's Guide*, SH12-6282
- *MERVA Family USE Administration Guide*, SH12-6065

MERVA OS/2 Books

- *MERVA OS/2 V3 and MERVA ESA V3 General Information*, GH12-6018
- *MERVA OS/2 V3 Licensed Program Specifications*, GH12-6057
- *MERVA OS/2 V3 Application Programming*, SH12-6058
- *MERVA OS/2 V3 Diagnosis Guide*, SH12-6059
- *MERVA OS/2 V3 User's Guide*, SH12-6060
- *MERVA OS/2 V3 Installation and Customization Guide*, SH12-6061

MERVA AIX Books

- *MERVA AIX Licensed Program Specifications*, GH12-6180
- *MERVA AIX User's Guide*, SH12-6181
- *MERVA AIX Installation and Customization Guide*, SH12-6182
- *MERVA AIX Application Programming*, SH12-6183
- *MERVA AIX Diagnosis Guide*, SH12-6184

MERVA ESA Books

- *MERVA OS/2 V3 and MERVA ESA V3 General Information*, GH12-6018
- *MERVA ESA V3 Licensed Program Specifications*, GH12-6019
- *MERVA ESA V3 Application Programming Interface Guide*, SH12-6183
- *MERVA ESA V3 Operations Guide*, SH12-6021
- *MERVA ESA V3 User's Guide*, SH12-6022
- *MERVA ESA V3 Macro Reference*, SH12-6023
- *MERVA ESA V3 Installation Guide*, SH12-6025
- *MERVA ESA V3 Messages and Codes*, SH12-6026
- *MERVA ESA V3 Customization Guide*, SH12-6027
- *MERVA ESA V3 Concepts and Components*, SH12-6028
- *MERVA ESA V3 Advanced MERVA Link*, LY12-5081
- *MERVA ESA V3 Workstation Based Functions*, SH12-6069
- *MERVA ESA V3 IFT Connection for MVS*, SH12-6280
- *MERVA ESA V3 Traffic Reconciliation Reference*, SH12-6281

Further IBM Publications

- *IBM OS/2 Information and Planning Guide*, G326-0160
- *IBM DATABASE 2 for OS/2 Planning Guide*, S20H-4784
- *OS/2 Programming Tools and Information Version 1.3 Control Program Programming Reference*, S91F-9260-00
- *IBM Communications Server Version 4.1 Up and Running !*, GC31-8189
- *IBM Personal Communications Version 4.1 for OS/2 Up and Running !*, GC31-8258
- *IBM AIX and Related Products Documentation Overview*, SC23-2456
- *IBM DATABASE 2 for AIX Planning Guide*, S20H-4758
- *CICS/ESA Intercommunication Guide*, SC33-0657
- *CICS/ESA System Definition Guide*, SC33-0664
- *CICS/ESA Resource Definition (Online)*, SC33-0666
- *CICS/ESA Resource Definition (Macro)*, SC33-0667
- *CICS/ESA Messages and Codes*, SC33-0672
- *CICS/ESA Application Programming Guide*, SC33-0675
- *CICS/ESA Application Programming Reference*, SC33-0676

S.W.I.F.T. Publications

The following books are published by the Society for Worldwide Interbank Financial Telecommunication, s.c., in La Hulpe, Belgium:

- *S.W.I.F.T. User Handbook(1996)*
- *S.W.I.F.T. Dictionary (1996)*
- *S.W.I.F.T. FIN Security Guide (1996)*
- *S.W.I.F.T. Card Readers User Guide (1996)*

Index

N

P

R

S

T

U

V

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

MERVA Family
MERVA Connection/ESA

Publication No. SH12-6187-00

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

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

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

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

Please tell us how we can improve this book:

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

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

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SH12-6187-00

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
Postfach 1380
71003 Boeblingen
Germany

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

SH12-6187-00

Cut or Fold
Along Line

Program Number: 5622-122 OS/2 LAN
5622-127 OS/2 Standalone
5765-449 AIX

Printed in Denmark by IBM Danmark A/S

SH12-6187-00