MERVA for ESA

Advanced MERVA Link

Version 4 Release 1

SH12-6390-01

MERVA for ESA

Advanced MERVA Link

Version 4 Release 1

SH12-6390-01

Note!
Before using this information and the product it supports, be sure to read the general information under "Appendix B. Notices" on page 171.

Second Edition, May 2001

This edition applies to

Version 4 Release 1 of IBM MERVA for ESA (5648-B29)

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

Changes to this edition are marked with a vertical bar.

Contents

About This Book

MERVA Link is a component of the IBM licensed program Message Entry and Routing with Interfaces to Various Applications for ESA Version 4 Release 1 (abbreviated to MERVA ESA). It provides a means to interconnect cooperating MERVA systems in an SNA or an IP network. This interconnection is used to exchange MERVA ESA messages between the cooperating MERVA systems (MERVA Link Message Handling System) and to control a partner MERVA system by means of sending a MERVA ESA command (MERVA System Control Facility). The MERVA System Control Facility is supported only in an SNA network.

This book contains the information necessary to understand the MERVA Link Message Handling System (MHS) architecture and implementation, and information that is not covered by other books of the MERVA ESA library.

Who This Book Is For

This book is for:

- Anyone who wants to become familiar with the concept of the MERVA Link and its terminology
- System programmers responsible for setting up a network of cooperating MERVA systems
- Application programmers responsible for developing their own implementation of the MERVA Link protocol
- Anyone who is responsible for analyzing problems in a network of cooperating MERVA systems.

What You Need to Know to Understand This Book

It is assumed throughout this book that you have experience with single MERVA ESA systems, and basic knowledge of SNA concepts and terminology as well as knowledge of the CICS and APPC/MVS intersystem communication facilities. If you plan to use the MERVA Link functions executing in the OS/390® UNIX System Services environment, you must have basic knowledge of the OS/390 UNIX System Services.

How to Use This Book

Read Chapter 1 to familiarize yourself with the concepts, the resources, and the terminology of MERVA Link.

Thereafter, you can use the appropriate parts of the book as guidance and reference material for your particular task.

Summary of Changes

This edition of this manual reflects the following differences between the current version of MERVA ESA (Version 4.1) and the previous version (Version 3.3):

What Has Been Added

MERVA Link UNIX System Services

MERVA ESA now provides a set of MERVA Link functions called MERVA Link USS. These functions execute in the OS/390 UNIX System Services (USS) environment, and provide gateway services for routing MERVA Link conversations from an SNA APPC network to a TCP/IP network, and vice versa.

What Has Been Modified

MERVA Link P1 and P2 Protocols

The definitions of the MERVA Link P1 and P2 protocols have been extended to provide additional functionality. The protocol definitions and PDU data element summary have been changed accordingly.

MERVA Link Conversation Trace Events

A number of CICS commands related to an APPC conversation have been added to the events shown in the MERVA Link conversation trace. The conversation trace description and the conversation trace samples have been changed accordingly.

What Has Been Removed

MERVA Link Asynchronous Communication

Support of the asynchronous communication protocol based upon LU 6.1 services has been dropped. The corresponding topics have been removed from this manual.

MERVA Link APPC/MVS Mirror

Support of the MERVA Link APPC/MVS Mirror has been dropped in MERVA Link. The corresponding topics have been removed from this manual.

MERVA Link Internals

The description of a number of MERVA Link internal implementation topics has been removed from this manual. These topics are:

- MTL and MTP boundary interfaces
- APPC and ISC boundary interfaces
- Internal module and service primitive traces
- Program return codes
- Storage areas in MERVA Link dumps
- Journal entry samples
- MERVA Link IMS APPC considerations
- MSC processing structure

Part 1. The MERVA Link Message Handling System

Chapter 1. Introduction

This chapter introduces the MERVA Link message handling system (MHS). It contains a description of the MERVA Link message handling system model, that is, MERVA Link's conceptual view of the world. It names the protocols defined and used in MERVA Link and introduces the MERVA Link implementation.

The Model of the MERVA Link Message Handling System

The MERVA Link message handling system model serves as a tool to aid in the development of an architecture for a message handling system and describes the basic concepts graphically. It consists of several different functional components that work together to provide message handling services. The model can be applied to a number of physical and organizational structures.

The message handling system model employed by MERVA Link is described in the following; and familiarity with it is necessary to fully understand the MERVA Link.

Functional View of the Model

The functional view of the MERVA Link message handling system model is shown in Figure 1.

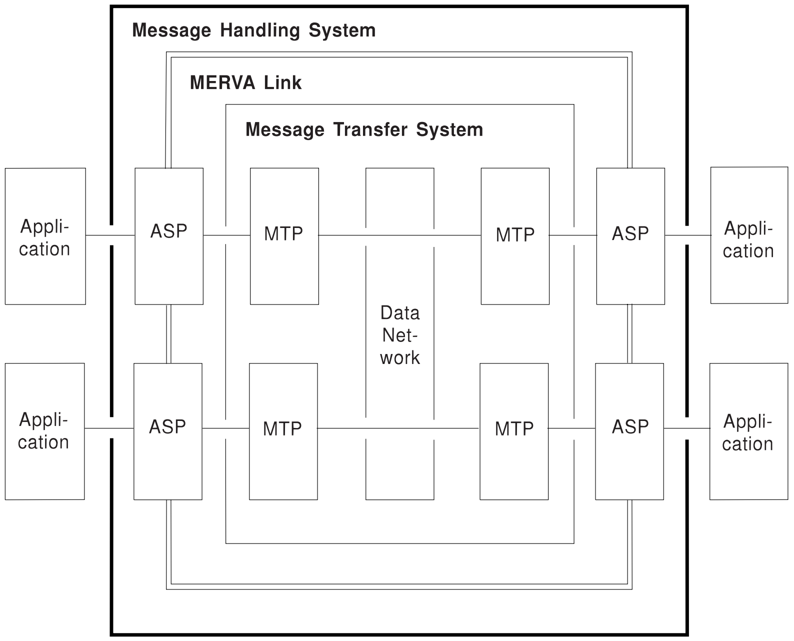

Figure 1. MERVA Link MHS Model: Functional View

In this model, an application is a computer process that requests message handling services from the message handling system (MHS). An application is referred to as either:

- An originating application (when preparing and sending a message)
- A recipient application (when receiving a message)

An originating application prepares a message and passes it to its application support process (ASP). An ASP is a computer process that interacts with the message transfer system (MTS) to submit messages for a single originating application. The MTS, as known by a specific ASP, consists of a sending and receiving message transfer process (MTP) and works as follows:

1. A sending ASP submits a message to its associated sending MTP.
2. The sending MTP transfers the message to its partner receiving MTP.
3. The receiving MTP delivers the message to its associated receiving ASP.
4. The receiving ASP passes the message to the recipient application associated with it.

The MERVA Link message transfer system (MTS) consists of a set of cooperating MTPs and a data communication network interconnecting cooperating MTPs. A MERVA Link P1 Routing Gateway can be part of the MTS. A MERVA Link Gateway can interconnect an MTP using SNA APPC with an MTP using TCP/IP.

Layered Representation of the Model

The layered representation of the MERVA Link MHS Model is shown in Figure 2. The explanation of the abbreviations used in this figure is contained in the text following that figure.

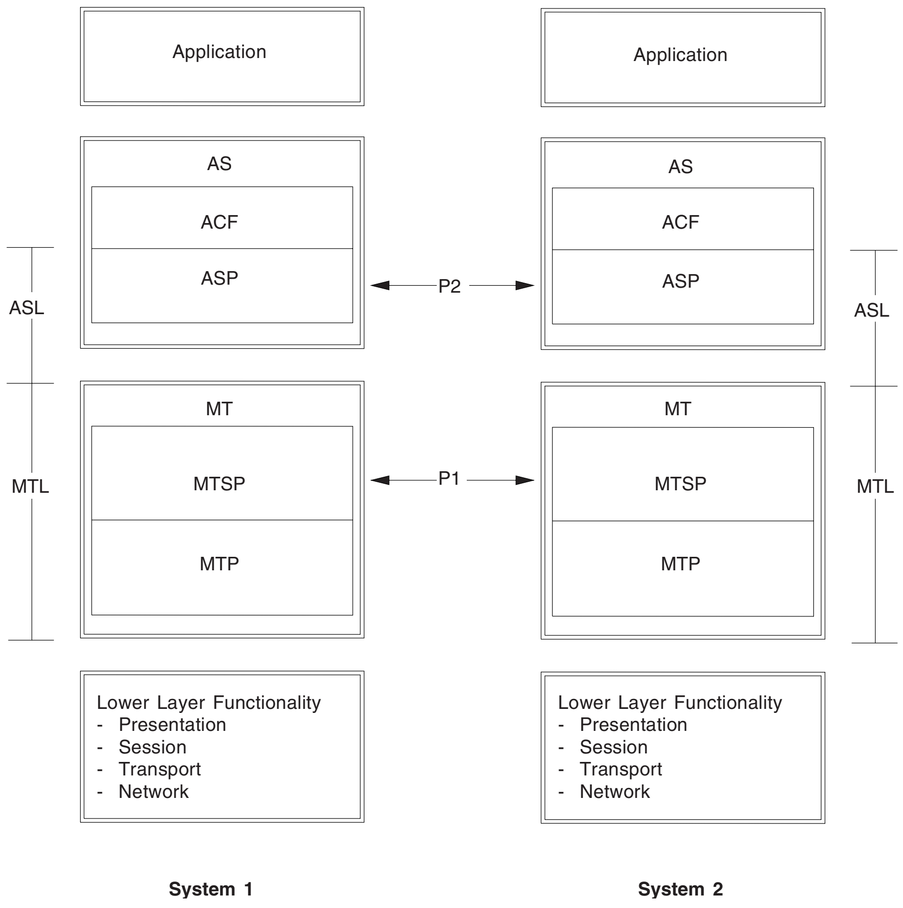

Figure 2. MERVA Link Message Handling System Model: Layered Representation

Figure 2 uses the following abbreviations:

AS MERVA Link Application Support is a set of functions that provide ASL and MTL services to applications. An executing entity of MERVA Link Application Support is called an application support process (ASP).

ACF	The application communication functionality (ACF) is contained in a sublayer between the application and the MERVA Link application support layer (ASL). It is closely related to the application. The functionality contained in this sublayer is specified individually by each application and is not part of a general MERVA Link application support service. Application communication functionality can be implemented in a MERVA Link user exit.
ASL	The application support layer (ASL) contains the general MERVA Link application support functionality. This functionality provides the application support services that are independent of a specific application.
ASP	A sending or a receiving application support process is an executing entity of MERVA Link Application Support. It supports a specific application. An ASP can be adapted to a specific application through individual application communication functionality (ACF) and through ASP customization. A sending ASP obtains a message from the application, converts it to the internal MERVA Link format, and submits the message to the message transfer system. When the message transfer system has delivered a message to a receiving ASP, the ASP converts it from the internal MERVA Link format to the application format and passes the message to the application.
MT	MERVA Link Message Transfer (MT) is a set of functions that provide message transfer services to MERVA Link Application Support. An executing entity of MERVA Link Message Transfer is called a message transfer process (MTP).
MTL	The message transfer layer (MTL) contains the MERVA Link message transfer functionality.
MTSP	The MERVA Link message transfer service processor (MTSP) contains the main message transfer layer functionality. It interfaces with all sending and receiving ASPs and all sending and receiving MTPs. The MTSP deals with the MERVA Link message transfer protocol, called P1. It is therefore also called the MERVA Link P1 processor. All functionality that deals with the actual transfer of a message is provided by a message transfer program.
MTP	A sending or a receiving message transfer process is an executing entity of MERVA Link Message Transfer. The functions of an MTP are implemented in a message transfer program, also called a message transfer processor. A MERVA Link message transfer program communicates with the message transfer service processor and sends or receives protocol data units (PDUs) to or from its partner. An MTP uses SNA APPC (LU 6.2) or TCP/IP Stream Socket services to communicate with its partner process.
P1	The peer-to-peer protocol at the message transfer layer is called P1. P1 defines the language used by cooperating message transfer processes.
P2	The peer-to-peer protocol at the application support layer is called P2. It

supports messaging between applications, called Interapplication Messaging (IAM). P2 defines the language used by cooperating IAM application support processes.

Boundary Functions

The model of the MERVA Link message handling system shows that MERVA Link is organized in layers and sublayers. Boundary functions are supported at the boundaries of all layers and sublayers.

MERVA Link Boundaries

The functions at the boundary between the application and MERVA Link is described by the MERVA ESA functions and the MERVA Link application support functions.

The functions at the boundaries between MERVA Link layers and sublayers are described by boundary function service primitives.

The functions at the boundary between a MERVA Link message transfer processor and the data communication services used by that MTP are specified by the corresponding data-communication system (CICS (R) or APPC/MVS), and by MERVA Link implementation rules.

Boundary Function Service Primitives

Service primitives can be compared with program function calls and the return to the calling program. Parameters are associated with the function call and with the return to the caller.

The name of a service primitive consists of two parts that are connected by a period. The first part denotes the service primitive function and is written in uppercase letters except for the two exceptions, SendPDU and ProcessPDU. Examples of names of service primitive functions are CONNECT, SUBMIT, DELIVER, and DISCONNECT.

The second part of a service primitive name identifies the type of the call. Types of service primitives are Request, Confirmation, Indication, and Response.

The call of an upper-layer program to a lower-layer program is called a *request*. A request is a primary service primitive. A *confirmation*, the return to the calling program, is associated with every request. A confirmation is a secondary service primitive.

The call of a lower-layer program to an upper-layer program is called an *indication*. An indication is a primary service primitive. A *response*, the return to the calling program, is associated with every indication. A response is a secondary service primitive. Notice that the upper-layer programs at the receiving side are called by the lower-layer programs.

Peer-to-Peer Protocols

The following peer-to-peer protocols are defined in MERVA Link:

P1 The message transfer protocol

P2 The application support protocol

P3 The command transfer protocol

Message Transfer and Application Support Protocols (P1 and P2)

At the interapplication messaging application support layer (IAM ASL), an application message consists of a message header and a message body. The message body contains a single body part (implementation rule). The body part consists of a body part header and body part data. The body part data consists of one or more body part data segments.

An acknowledgment message at the application support layer contains only status information in a status report.

At the message transfer layer, a message consists of a message envelope, its content (optional), and a message trailer. The content of an application message is an ASL message (consisting of a message header and a message body, or a status
| report). A service message at the message transfer layer can be a test message
| (probe). The support of a MERVA Link delivery report has been dropped. A test message consists only of a PROBE envelope and the message trailer.

The MERVA Link message structure at its highest level is shown in Figure 3.

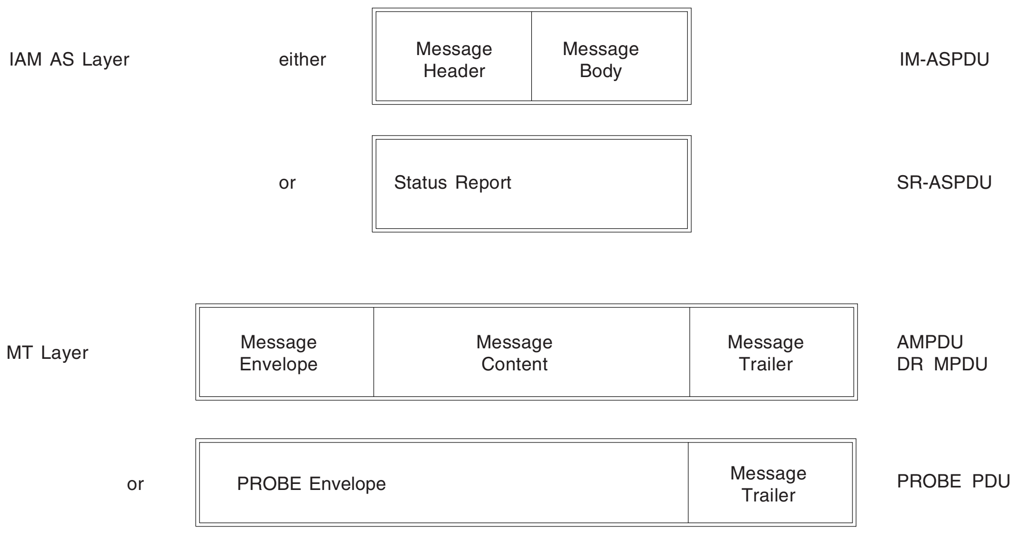

Figure 3. MERVA Link Message Structure

The messages exchanged between MERVA Link peer entities (cooperating ASPs and MTPs) are represented by protocol data units (PDUs). The encoding of MERVA Link P1 and P2 PDUs corresponds to the format of an SNA LU 6.2 general data stream (GDS). The main characteristics of this data stream are its sequence of data elements and the format of a data element.

A data element consists of a 4-byte prefix (2-byte data-element length and 2-byte data-element identifier) and the data-element data (optional). Data-element data can consist of one or more data elements.

A complete description of MERVA Link P1 and P2 is in “Chapter 3. Peer-to-Peer Protocols” on page 29.

Command Transfer Protocol (P3)

The MERVA Link Command Transfer protocol (P3) covers the requirements of a partner system control function. It defines structures (PDUs) that are similar to the PDUs of the P1 and P2 protocols.

A command request PDU and a command response PDU consist of an envelope, a content, and a trailer. It is the same structure as defined by P1. A command request content consists of a command request heading. A command response content, however, consists of a command response heading and an optional command response body.

A complete description of MERVA Link P3 is in “Chapter 3. Peer-to-Peer Protocols” on page 29.

Implementation Overview

Two implementations of MERVA Link functions are available from MERVA ESA. MERVA Link ESA executes in the MERVA ESA CICS and IMS (TM) environments. MERVA Link USS executes in an OS/390 UNIX System Services environment and provides MERVA Link gateway services independently of a MERVA ESA installation.

MERVA Link ESA Implementation Overview

The main elements of the MERVA Link ESA implementation are introduced in the following.

MERVA Link Partner Table

The MERVA Link partner table (PT) is the main MERVA Link control resource. It contains static control information as well as dynamic status information. The PT is the means to customize MERVA Link.

The MERVA Link PT consists of a PT header and a number of PT entries of different types. It is generated by coding a set of EKAPT macro instructions:

- EKAPT TYPE=INITIAL generates the table header. It contains MERVA Link control information that is unique in a MERVA Link system.
- EKAPT TYPE=ASP generates an ASP entry. An ASP entry contains MERVA Link control information concerning a specific message transfer application.
- EKAPT TYPE=MTP generates an MTP entry. An MTP entry contains MERVA Link control information concerning a specific message transfer communication partner.
- EKAPT TYPE=SCP generates an SCP entry. An SCP entry contains MERVA Link control information concerning a specific system control communication partner.

MERVA Link Application Support Program

An Application Support Program (ASP) is a program that serves as the interface between an application and the message transfer system. The application either generates outgoing messages or processes incoming messages. The characteristics of the application are respected in this program.

For outgoing messages, an ASP is scheduled by the application program. It receives the message and control information from the application. Defaults apply to support outgoing messages without or with only a minimum of control information.

Using this control information and information from the applicable MERVA Link PT entry, a message heading and a SUBMIT.Request is generated. All message text and control data is passed to the application support filter, if applicable, or directly to the message transfer service processor. The message transfer service processor or the ASF returns to the ASP with a SUBMIT.Confirmation. Finally, the ASP processes the outgoing message as specified by the applicable MERVA Link PT entry. This final message processing can be one of the following:

- Delete it from the input MERVA ESA queue
- Delete it from the input MERVA ESA queue and put it to another queue
- Delete it from the input MERVA ESA queue and request routing by a MERVA ESA routing table

For incoming messages and reports, an ASP or an ASF is scheduled by the message transfer service processor. The latter processor was scheduled by the MTP that received the message from the remote system. The ASP passes the message and control information obtained in a DELIVER.Indication to the application as specified by the applicable MERVA Link PT entry.

MERVA Link Application Support Filter

A MERVA Link application support filter (ASF) is a program similar to a user exit. A user exit is normally called by a standard program and returns control to this standard program as soon as it has performed some control functions that are unique in a specific user environment. Standard processing is then continued by the standard program.

A MERVA Link ASF performs a similar kind of user-specific control function. However, its link to standard programs is different. It is inserted between two standard programs rather than called by one standard program. The entry and exit interface of an ASF (ASF upper-layer boundary interface) is the same as the ASF lower-layer boundary interface, and matches the interface between the standard programs. You can insert any number of application support filters between two standard programs.

The difference between a user exit and a MERVA Link ASF is shown in Figure 4.

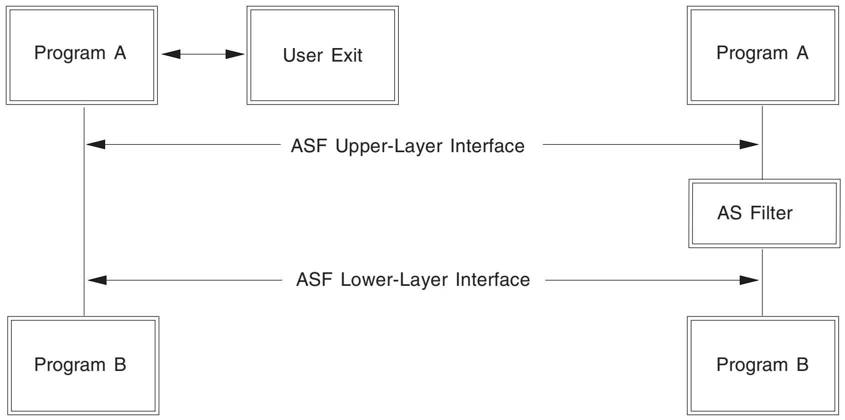

Figure 4. MERVA Link ASF Compared with a User Exit

For events originating at the upper-layer (for example, a SUBMIT.Request) each ASF must know the name of its lower-layer ASF or the name of the lower-layer standard program, and it must call that program. When this program has returned control to the ASF, the ASF can again perform some control functions and, finally, return control to its caller.

For events originating at the lower-layer (for example, a DELIVER.Indication) each ASF must know the name of its upper-layer ASF or the name of the upper-layer standard program, and it must call that program. When this program returns control to the ASF, the ASF can again perform some control functions and, finally, return control to its caller.

A MERVA Link ASP calls the first ASF in a list of ASFs. The message transfer service processor calls the last ASF in a list of ASFs.

MERVA Link Service Primitive Filter

MERVA Link supports two classes of ASFs. The first class, which keeps the name ASF, comprises the filters that get control for service primitives that handle an application message. These service primitives are the SUBMIT.Request and the DELIVER.Indication. An ASF in that class is not called for any other service primitive crossing the MTL boundary (for example, a CONNECT.Request or a TEST.Indication).

The second class of ASFs comprises the Service Primitive Filters. The only difference between an ASF and a Service Primitive Filter (SPF) is that the latter filters are called for all service primitives crossing the MTL boundary.

MERVA Link Message Transfer Service Processor

The message transfer service processor (MTSP) contains the central functionality of the MERVA Link message transfer system. It communicates with the upper-layer functions via the MERVA Link message transfer services interface. Some of the tasks to be performed by the MTSP are:

- Validate input parameters including a formal check of the MERVA Link P2 data stream for a SUBMIT.Request
- Find the name of the applicable message transfer process for a SUBMIT.Request
- Generate a unique message transfer layer message identifier for a submitted message
- Generate or analyze the message envelope according to the MERVA Link P1 protocol
- Call the applicable message transfer processor
- Return with a SUBMIT.Confirmation to the calling ASF or ASP as a response to a submitted message
- Pass an incoming message or an incoming receipt report to the applicable ASF or ASP

MERVA Link Message Transfer Processor

A message transfer processor (MTP) is a program that transfers a message from the local system to a specific remote system, or receives a message from a specific remote system. The communication protocol bilaterally agreed upon with the partner MTP is respected in this program.

For outgoing messages, an MTP is scheduled by the message transfer service processor. The message text and control information is passed to the MTP in the main storage. A conversation with the partner is established and the message is transferred to the partner.

For incoming messages, an MTP is scheduled by the applicable data communication system. One or more messages are received from the partner, and all messages are passed, message-by-message, to the MTSP.

MERVA System Control Facility

An interactive control facility is provided by MERVA Link to issue commands for the complete MERVA system including MERVA Link. The complete MERVA system includes also all partner MERVA ESA systems that are interconnected with the local MERVA system by a MERVA Link APPC connection. A partner MERVA AIX® system supports a subset of the MERVA Link commands supported by the MERVA System Control Facility.

The MERVA System Control Facility is the means to issue MERVA ESA commands with the same functionality that is provided by the MERVA ESA CMD function.

The MERVA System Control Facility is the means to display MERVA Link customization parameters, supervise the execution of MERVA Link, and to modify the execution parameters of MERVA Link. It is also the means to display status and error information that is collected by the MERVA Link CICS programs in the partner table.

A detailed description of the MERVA System Control Facility can be found in the *MERVA for ESA Operations Guide*.

The MERVA System Control Facility cannot be used to perform MERVA ESA user maintenance (function USR) and authenticator-key file maintenance (function AUT) in a partner MERVA ESA system.

MERVA Link USS Implementation Overview

The main elements of the MERVA Link USS implementation are introduced in the following.

Application Control Table (ACT) and Application Control Daemon (ACD)

The MERVA Link USS application control table (ACT), an OS/390 USS shared memory region, is the main MERVA Link USS control resource. It contains static control information as well as dynamic status information. The ACT is the means to customize MERVA Link USS.

The MERVA Link USS application control daemon (ACD), a long running OS/390 USS background process, is the owner of the ACT. The ACD creates the ACT based on information in MERVA Link USS configuration and security files.

The ACT consists of an ACT header, a number of ASP entries, and a number of ISC entries:

- The ACT header contains information that is unique for the MERVA Link USS instance represented by this ACT, for example the MERVA Link local node name.
- An ACT ASP entry contains information that applies to a specific MERVA Link USS ASP and its partner ASP in a partner MERVA Link system. Application Support functions are, however, available from MERVA Link USS for installation verification purposes only. MERVA Link USS does not provide functions to deliver a message to a MERVA ESA application.

- An ACT ISC entry contains intersystem connection information that applies to a specific partner node. It can be information for an SNA APPC and a TCP/IP connection to the partner system that houses the subject partner MERVA Link node.

AS Layer Programs

The MERVA Link AS layer contains the AS and P2 sublayers. The outbound and inbound functions of both sublayers are implemented in four separate programs:

ekaaso contains the outbound AS sublayer functions. Program ekaaso is available in a test version only. It cannot be used to handle messages.

ekaasi contains the inbound AS sublayer functions. Program ekaasi is available in a test version only. It prints information contained in a TEST or DELIVERY indication to a processing trace file. Program ekaasi cannot deliver an inbound message to MERVA ESA.

ekap2o contains the outbound P2 sublayer functions. Program ekap2o is available in a test version only.

ekap2i contains the inbound P2 sublayer functions. Program ekap2i is available in a test version only.

MT Layer Programs

The MERVA Link MT layer contains the P1 and MT sublayers. The outbound and inbound functions of both sublayers are implemented in six separate programs:

ekap1o
contains the outbound P1 sublayer functions. Program ekap1o is available in a test version only. It cannot be used to handle messages.

ekap1i
contains the inbound P1 sublayer functions. Functions of the outbound P1 sublayer are part of this program. These functions provide the P1 Routing service element, the main application function of MERVA Link USS.

ekatpo
contains the outbound MT sublayer functions for an SNA APPC connection to a partner MERVA Link system.

ekatpi contains the inbound MT sublayer functions for an SNA APPC connection from a partner MERVA Link system.

ekatco contains the outbound MT sublayer functions for a TCP/IP connection to a partner MERVA Link system.

ekatci contains the inbound MT sublayer functions for a TCP/IP connection from a partner MERVA Link system.

Application Control Programs

MERVA Link USS provides a number of programs that are used to operate and control MERVA Link USS functions:

ekaacc is the MERVA Link USS application control command application (ACC). It provides an interactive USS shell command interface to control and modify MERVA Link USS functions.

ekaacs is the MERVA Link USS local security control application (ACS). It provides an interactive USS shell command interface to specify security information for use by an outbound TCP/IP process.

Configuration Verification

Use a MERVA Link USS configuration verification environment to verify the correctness of a configuration without affecting an active MERVA Link USS node. For each of the programs **ekaacc**, **ekaacd**, and **ekaacs**, there is a corresponding program (**ekavcc**, **ekavcd**, and **ekavcs**) for use in a configuration verification environment.

Chapter 2. Service Elements

This chapter describes the service elements provided by the MERVA Link Message Transfer and Application Support services. It contains also detailed descriptions of important MERVA Link service elements, the Message Integrity Protocol (provided by MERVA Link ESA AS) and the P1 Routing service (provided by MERVA Link USS MT).

The Message Transfer Service

The message transfer service lets application support functions access and be accessed by the message transfer system in order to exchange messages.

The features of the message transfer service are:

Content Type Indication

This service element enables an originating ASP to indicate the content type for each submitted message.

The only content type supported by MERVA Link is MERVA Link P2 (Inter-Application Messaging).

MTL Message Identification

This service element enables the MTS to provide an ASP with a unique identifier for each message submitted to or delivered by the MTS. Application support functions and the MTS use this identifier to refer to a previously submitted message in connection with the delivery-notification service element.

The MTSP generates the MTL message identifier as the character representation of the S/390 (R) system clock.

ASL Content Identification

This service element enables the ASP to provide the MTS with an identifier for each message submitted to or delivered by the MTS. Application support functions can use this identifier to refer to a previously submitted message in connection with the delivery-notification service element.

Encoded Information Type Indication

This service element enables an originating ASP to specify to the MTS the encoded information type of a message being submitted. When a message is delivered, it also indicates to the recipient ASP the encoded information type of the message specified by the originating ASP.

Encoded information types are MERVA ESA queue format and MERVA ESA network (line) format.

Submission Timestamp Indication

This service element enables the MTS to indicate to an originating ASP and to the recipient ASP the date and time at which a message was submitted to the MTS. The format of the timestamp is *YYMMDDHHMMSS*.

Partner Connection

This service element enables an originating ASP to acquire a connection to a partner ASP and to test for its availability. This explicitly acquired connection can be used by the ASP to transfer a set of messages to this partner ASP.

Grade of Delivery Indication (Priority)

This service element enables an originating ASP to specify to the MTS the priority of the message. When a message is delivered, the message priority is indicated to the recipient ASP.

This service element does not support a message transfer with different priorities.

P1 Routing

This service element provides for routing an inbound SNA APPC or TCP/IP conversation to an outbound SNA APPC or TCP/IP conversation at the MERVA Link P1 sublayer. A detailed description of the P1 Routing service of MERVA Link USS can be found in “P1 Routing Facility” on page 26.

The Application Support Service

The Application Support Service, which is built upon the Message Transfer Service, is provided by the class of application support functionalities supporting the MERVA Link Inter-Application Messaging (P2) protocol. This service enables an application to send a message to a recipient and to have it received by this recipient. Application Support Functionalities make use of the Message Transfer Services of the MTS described in “The Message Transfer Service” on page 15. In addition, application support functionalities supporting the MERVA Link P2 protocol provide other capabilities that are elements of the Application Support Service. For example, they uniquely identify each message and the nature and attributes of its body.

The features of the Application Support Service are:

Message Transfer Service Elements

The service elements that are part of the message transfer service and that are used to transfer an application message are available as part of the application support service.

ASL Message Identification

This service element permits cooperating ASPs to convey an identifier for each application message sent or received. The applications and the ASPs use this identifier to refer to a previously sent or received message (for example, in a Receipt Confirmation).

The ASP generates the IAM message ID as the character representation of the S/390 system clock if this message identifier is not already contained in the message or if it is not provided by the ACF.

Typed Body

This service element permits the nature and attributes of the body of the application message to be conveyed together with the body. Any MERVA Link defined body type can be designated.

MERVA Link defined body types are MERVA ESA Network (line) format and MERVA ESA queue format. The attributes of a body in MERVA ESA line format are, for example, the MERVA ESA message type (MCB identifier) and the line identifier.

Receipt Confirmation

This service element allows the originator to request that he be notified that an application message was received or not received by the intended recipient application. The actual meaning of the receipt or nonreceipt event is defined by the cooperating applications.

Acknowledgment Message Correlation

This service element allows the originator to request that an incoming acknowledgment message (also called receipt report or status report) is correlated with the reported message and the receipt information is merged with the reported message.

Originator/Recipient Indication

This service element allows the identity of the originating and recipient application to be conveyed to the recipient application. The MTS provides to the recipient ASP the authenticated address of the originator and the recipient. In contrast, the intention of this AS Service element is to identify the originator and the recipient application in terms specified by the cooperating applications.

Subject Indication

This service element enables the originating application to indicate to the recipient application the subject of the application message being sent. The subject information is made available to the recipient application.

A subject is conveyed to the recipient application if it is contained in the MERVA ESA message.

The maximum length of a subject is 60 characters.

Buckslip (Attached Note)

This service element enables the originating application to attach a short note to the application message being sent. The buckslip is made available to the recipient application.

A buckslip is conveyed to the recipient application if it is contained in the MERVA ESA message. The maximum length of a buckslip is 256 characters.

Message Integrity Protocol

The Message Integrity Protocol (MIP) service element provides functionality that ensures that:

- Any loss of a message that was in a MERVA Link send queue and was processed by MERVA Link is detected and reported.
- No message is passed twice to the receiving application, regardless of what kind of system failure occured, and regardless of at which point in the message transfer process it occurred.

Message Text Encryption Indication

This service element allows the originator to indicate to the recipient that the text of the application message being sent is encrypted. An identification of the encryption algorithm or the encryption key, or both, can be conveyed to the recipient. This service element can be used by the recipient to determine that the text of the application message must be decrypted.

This service element does not encrypt or decrypt the message text. The encryption and decryption process can be performed in an application support filter, a program similar to a user exit.

The identifier of the body part data segment tells whether that part of the message has actually been encrypted (ID=8123) or has not been encrypted (ID=8122).

Message Text Encryption

A specific message-encryption service is provided by MERVA Link. This service element is provided using the message-encryption indication service element and a MERVA Link proprietary encryption algorithm.

Message Text Authentication Indication

This service element allows the originator to convey an authentication key and the identification of an authentication algorithm to the recipient.

Message authentication means protecting the message against unauthorized modification during the transfer process. The authentication process does not encrypt the message.

This service element does not authenticate the message text. The authentication process can be performed in an application support filter, a program similar to a user exit.

Message Text Authentication

A specific message-authentication service is provided by MERVA Link. This service element is provided using the message-authentication Indication service element and a MERVA Link proprietary authentication algorithm.

Message Text Compression Indication

This service element allows the originator to convey a compression key and the identification of a compression algorithm to the recipient.

This service element does not compress the message text. The compression/expansion processes can be performed in an application support filter, a program similar to a user exit.

Priority Indication

This service element allows the originator to convey information about the priority of the message to the receiver. It does not provide priority handling of a message within the message transfer system.

Possible Duplicate Message Indication

This service element allows the originator to convey the information to the receiver that the message might have been transmitted through another medium (such as telex) in addition to the transmission through MERVA Link.

Body Part

This service element allows the originator to send to a recipient an application message with a body that consists of a single body part. This body part can consist of one or more body part data segments.

The MERVA Link Message Integrity Protocol (MIP)

Provisions are made in MERVA Link to ensure that no message passed to MERVA Link in a member of a MERVA Link send queue cluster is lost and that messages are not passed twice to the receiving application. MERVA Link ensures that messages transmitted twice are not passed twice to the receiving application despite possible system failures at any point of the message transfer process.

The MERVA Link Message Integrity Protocol (MIP) implementation is based on MERVA ESA Queue Management Services (QMS).

Message Integrity Protocol Terms

The descriptions of the MERVA Link Message Integrity Protocol (MIP) in the following sections use the following terms:

application control queue
The MERVA Link MIP uses a MERVA ESA queue to store:

- The last confirmed control message (LC MS)
- The last received control message (LR MS)
- The in-process messages (IP MS)

Every ASP defined in the partner table has its own application control queue (ACQ). A control queue cannot be shared between ASPs.

The application control queue should not be accessed by any MERVA ESA user directly. Information from control messages in the application control queue can be displayed using the MERVA System Control Facility (MSC).

last confirmed (LC) control message
The last confirmed (LC) control message is a message in the application control queue. It is owned by MERVA Link. This means, its content and format is specified by MERVA Link, and it is automatically generated by MERVA Link if necessary.

The main purpose of the LC control message is to monitor the status of the message transfer from the applicable ASP to its partner ASP (only in this direction). MERVA Link reports the status or any malfunction of the message transfer in the LC control message. This information can be displayed using the MERVA System Control Facility (MSC).

An LC control message is in an application control queue when a message has been sent by the applicable ASP to its partner.

in-process (IP) message

| An in-process (IP) message is a message being transferred to the partner
| ASP. An MIP message identifier and an MIP message sequence number

have been assigned to an IP message. The IP message remains in the application control queue until the message transfer has been confirmed by the message transfer service.

The maximum number of IP messages in an application control queue is equal to the static MIP window size. In an inactive system, no IP message should be in this queue. An IP MS in the application control queue indicates an exceptional condition or a malfunction if the system is inactive.

IP messages are subject to automatic recovery as specified by the MIP. IP messages are subject to automatic or manual recovery as specified by the MERVA Link **recover** and **iprecov** commands.

last received (LR) control message
The last received (LR) control message is a copy of the last message received by an ASP from its partner in the application control queue. It remains on this queue until another message has been received and been accepted by the receiving ASP. There is only one LR control message in an application control queue.

MIP message identifier
The MIP message identifier is a unique identifier generated by a sending ASP and associated by that ASP to an IP message. The content of this identifier is an 8-byte binary number with an increasing value each time it is generated. The MIP message identifier is used by the receiving ASP to determine whether an IP message is a new message that was generated after the MIP control information of the sending ASP was deleted.

MIP message sequence number (MSN)
The MIP message sequence number (MSN) is a wrap-around number between 1 and 9999 inclusive. For the next MSN following the number 9999, the MSN is wrapped around to the number 1. It is used by the sending and receiving ASP to verify that no message is missing and to detect duplicate messages.

static MIP window size
The static MIP window size specifies the maximum number of messages an ASP can submit before it must request a confirmation of the transfer process. An ASP must stop submitting messages until the requested confirmation has been received. That confirmation applies to all messages in this window.

dynamic MIP window size
The dynamic MIP window size specifies the index of a message in a window. The dynamic MIP window size of the first message in a window is 1. The dynamic MIP window size of the last message in a complete window is equal to the static MIP window size.

Description of the MIP at the Sending Side

The sending MERVA Link resources are shown in Figure 5 on page 21. The indicated contents of the application control queue reflect a normal processing status where the second message within a window was transferred to the partner MTP. The corresponding status at the receiving side is shown in Figure 6 on page 23.

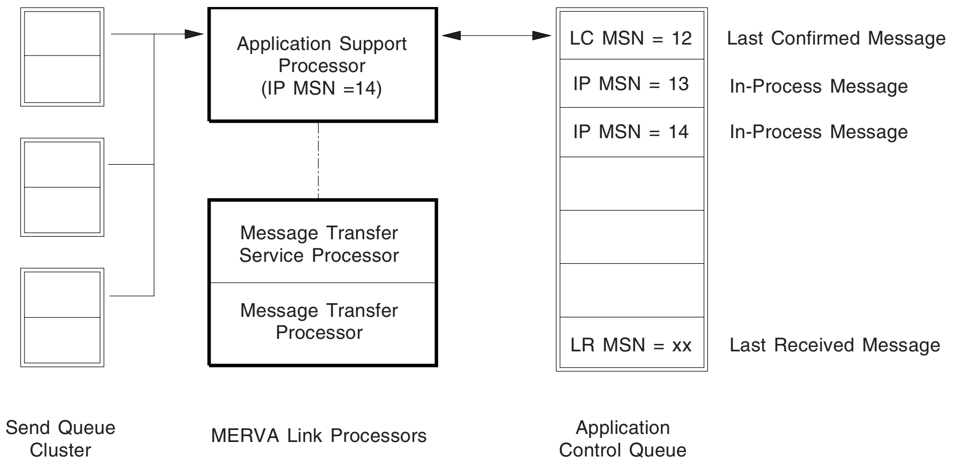

Figure 5. MIP Resources at the Sending Side

The MIP processing logic in the sending system is explained in the following.

The Send Queue Cluster

The sending MERVA ESA application is represented by a set of MERVA ESA queues that are categorized, for example, as urgent, normal, and low priority queues. When the local MERVA ESA application puts a message into one of these queues, the MERVA Link sending Application Support Processor is started. It identifies the applicable ASP entry in the PT and ensures that no other task works for this ASP. If there is already another task active for this ASP, it terminates immediately.

Begin Initial Message Processing

After connecting successfully to the partner system, the ASP checks whether the previous conversation was successfully completed. This is the status when there are no IP messages in the application control queue. Normal message processing begins in this case.

If there is no LC control message in the application control queue, an LC control message is generated with an LC MSN of 9999. The first message will therefore get the MSN 0001. A missing LC MS means that the ASP is started the first time. An LC MSN of 0000 means that an MIP reset has been requested by a MERVA Link administrator (MSC command **lcreset**).

Check the Integrity of IP Messages

All IP messages in the application control queue with an IP MSN not greater than the LC MSN are routed as specified by the MERVA ESA routing table associated with the application control queue. These messages were confirmed but routing of all or part of the confirmed messages failed.

All IP messages in the application control queue with an IP MSN that is greater than the LC MSN are recovered. This means they are sent to the partner system and confirmation is requested when the last IP message has been sent.

When a confirmation has been received from the partner, the LC MSN in the application control queue is updated with the last IP MSN. All IP messages are routed as specified by the MERVA ESA routing table associated with the application control queue after a confirmation.

When the partner process has signaled an error (nonconfirmation), the application status is set to 09 or 13 as response to the reported event return codes 08 (error) or 12 (severe error), respectively, the IP messages are not routed, and the ASP is terminated without processing any message in the send queue cluster.

For details, see “Message Integrity Checks at the Sending Side” on page 24.

Begin Normal Message Processing

Normal message processing according to the MIP is started at this point. The status is as follows:

- There is an LC control message in the application control queue containing an LC MSN.
- There are no IP messages in the application control queue.
- There is at least one message in one queue of the send queue cluster.

Normal message processing is initialized by defining a last-used message sequence number (LU MSN) in the ASP working storage. Its initial value is the LC MSN.

Now the ASP starts to process the messages in the send queue cluster in the sequence as specified by the priority of the individual queues. This means that the messages in the first queue in the cluster are processed first. The next message in the second queue of the cluster is processed if the first queue is empty. The next message in the third queue of the cluster is processed if the first and the second queue are empty.

Assign Message Sequence Number and Message Identifier

The ASP calculates the next message sequence number (MSN) and generates a unique MIP message identifier. Both values are stored in MERVA Link control fields of the IP message. The static MIP window size is also stored in a MERVA Link control field of the IP message.

Move IP Message to the Control Queue

The IP message, including new MERVA Link control information, is moved from the send queue to the application control queue. The message in the send queue was deleted in the “move” process (MERVA ESA Queue Management PUT with automatic delete from the source queue).

Assign Dynamic MIP Window Size

The ASP calculates the index of the IP message in the current window and includes it as the dynamic MIP window size in the message heading (P2 PDU).

Submit the Message and Handle the Submit Confirmation

The ASP now submits the message to the MTS and requests a message transfer confirmation if a MIP window boundary has been reached.

The MTS returns with a SUBMIT.Confirmation containing one of the indicators *accepted*, *confirmed*, or *error*. The ASP receiving the accepted indicator is asked to send the next message. In case of an error indicator it is asked to handle (report) that error and to terminate. If the ASP receives the confirmed indicator it is asked to synchronize its message transfer control status with the partner ASP.

To synchronize its message transfer control status with the partner ASP, the local ASP updates the LC MSN (last confirmed message number) with the LU MSN (last used message number) in the LC control message in the application control queue and routes all IP messages as specified by the MERVA ESA routing table associated with the application control queue.

No More Messages to Be Transferred

When all messages in the send queue cluster have been processed (queue empty condition on all queues of that cluster) the messages within the last window might not yet have been confirmed. In this case (LC MSN not equal LU MSN) a DISCONNECT.Request with the confirmed indicator is issued. If the last message was already confirmed, a DISCONNECT.Request without the confirmed indicator is issued.

Description of the MIP at the Receiving Side

The MERVA Link MIP resources at the receiving side are shown in Figure 6. The indicated contents of the application control queue reflect a normal processing status where the second message within a window was received from the partner MTP but not yet passed to the ASP. The corresponding status on the sending side is shown in Figure 5 on page 21.

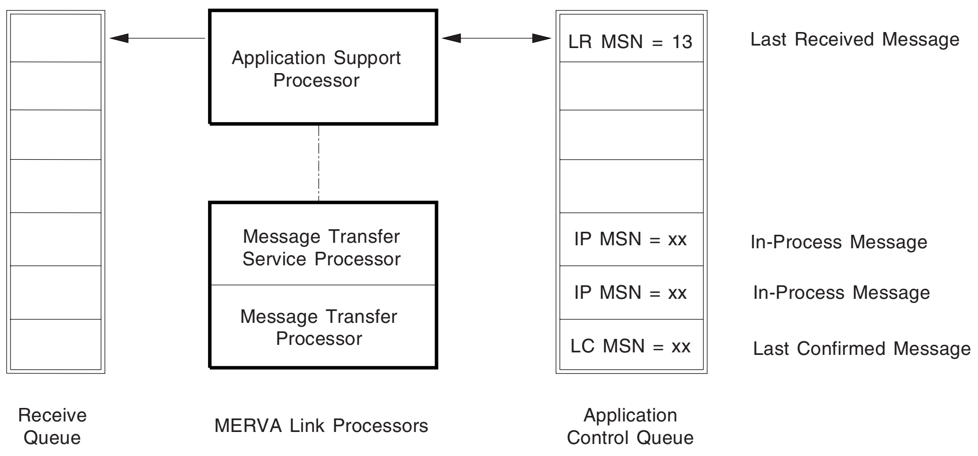

Figure 6. MIP Resources at the Receiving Side

The MIP processing logic in the receiving system is explained in the following.

Receiving ASP

The receiving side is represented by a message transfer process associated with an ASP and a MERVA ESA receive queue via MERVA Link customization in the partner table.

The MTP receives a complete message and passes it via the MTSP to its associated receiving ASP.

The receiving ASP analyzes the message and extracts the In-Process message sequence number (IP MSN), the MIP message identifier, the index of the IP message in its window, and the (optional) MIP reset indicator from the message heading.

Begin Inital Message Processing

The receiving ASP retrieves the "last received message sequence number" (LR MSN) and the dynamic MIP window size from the LR control message (LR MS) in the application control queue. If the MIP reset flag is set in the received message, the message integrity check is skipped without indicating an error condition. It is also skipped if there is no LR control message in the application control queue.

Check the Integrity of IP Messages

If the received message is the next expected message, it is delivered to the receiving application. An IP message that is outside the normal message sequence causes an implicit reset of the MIP if the following conditions apply:

- The IP MSN is 1.
- The IP message is the first message in its window.
- The MIP message identifier of the IP message is greater than the MIP message identifier of the LR control message.

Implicit MIP reset means that the LR control message is disregarded and the received message is delivered to the receiving application. This situation can occur, for example, when the MERVA ESA queue data set of the sending partner system is redefined or when all messages in the control queue of the partner ASP are deleted.

If the IP MSN is greater than the LR MSN plus one, an MIP violation is reported. If the IP MSN is less than the LR MSN minus the dynamic MIP window size contained in the LR control message, an MIP violation is also reported.

If the above conditions do not apply, the IP message is a member of a recovered window. It is considered as already received and delivered to the receiving application. The delivery confirmation was, however, not received or processed by the sending ASP. The received IP message is discarded without further processing. It will be confirmed later.

For details, see "Message Integrity Checks at the Receiving Side" on page 25.

Route Incoming Message

The ASP puts the received message in one MERVA ESA QMG ROUTE request to both the receive queues and the application control queue (as new LR control message), and automatically deletes the old LR control message from that queue. The MERVA ESA routing table that controls this process can be associated with any queue. It must ensure that the message is routed to the applicable application control queue in addition to the receive queue or queues.

When all AS processing has been successfully completed, the ASP returns to the MTSP and to the MTP, indicating successful completion.

Message Integrity Checks at the Sending Side

The message integrity check at the sending side is based on the following information:

LC	Message sequence number of the last confirmed message as contained in the LC control message in the application control queue.
IP	Message sequence number of the message currently in process as contained in the IP message in the application control queue.
WSZ	Static MIP window size as contained in the LC control message in the application control queue.

The message integrity check algorithm uses two variables, the first called the distance (D1) and the second called the wrap-around distance (D2) of the IP message from the LC message.

The distance D1 is defined as **`D1 = IP - LC.`**

The wrap-around distance D2 is defined as:

```
D2 = D1 if |D1| ≤ WSZ,
D2 = D1 - 9999 if D1 > 0, otherwise,
D2 = D1 + 9999
```

The rules must be applied in this sequence.

The action to be performed with the message currently in process is determined by the following rules:

```
IF D2 = WSZ THEN  send message,
IF |D2| ≥ WSZ THEN  indicate MIP violation,
IF D2 ≤ 0 THEN  route confirmed message locally,
OTHERWISE send message
```

The rules must be applied in this sequence.

An example of the action performed for various IP messages in the application control queue with an LC MSN of 15 and an MIP window size of 3 is shown in Table 1.

Table 1. MIP Check for LC MSN = 15 and WSZ = 3

IP MSN	10	11	12	13	14	**15**	16	17	18	19	20
D1 **D2**	-5 9994	-4 9995	-3 -3	-2 -2	-1 -1	0 0	1 1	2 2	3 3	4 -9995	5 -9994
Action	Protocol Error			Route MSG			Send MSG			Protocol Error	

Message Integrity Checks at the Receiving Side

A message integrity check at the receiving side is performed if an LR control message is available in the application control queue. This check is based upon the following information:

LR	Message sequence number of the last received message as contained in the LR control message in the application control queue.
LRID	MIP message identifier of the LR control message.
IP	Message sequence number of the message currently in process as contained in the received message.
IPID	MIP message identifier of the IP message.
DWSZ	Dynamic MIP window size as contained in the LR control message in the application control queue.
IPWX	Index of the IP message in the current window as contained in the MIP window size control field of the IP message.

The message integrity check algorithm uses two variables. The first variable is called the distance of the IP message from the LR control message (D1). The second variable is called the wrap-around distance of the IP message from the LR control message (D2).

The distance D1 is defined as **D1 = IP - LR.**

The wrap-around distance D2 is defined as:

```
D2 = D1 if |D1| ≤ DWSZ,
D2 = D1 - 9999 if D1 > 0, otherwise,
D2 = D1 + 9999
```

The rules must be applied in this sequence.

The action to be performed with the IP message is determined by the following rules, which must be applied in this sequence.

```
IF D2 = 1 THEN  process message (normal situation)
IF IP = 1, IPWX = 1, AND IPID > LRID THEN process message with implicit LRRESET
IF D2 > 1 THEN  report MIP violation
IF |D2| ≥ DWSZ THEN  report MIP violation
OTHERWISE discard message, it has already been delivered
```

The implicit reset of the LR control message is not executed if either of the MIP message identifiers (IPID or LRID) is not available. An MIP message identifier may not be available as the corresponding data element in a P2 PDU is defined as optional.

An example of the action performed for various IP messages with an LR MSN of 14 and a dynamic MIP window size of 3 in the LR control message is shown in Table 2.

Table 2. MIP Check for LR MSN = 14 and DWSZ = 3

IP MSN	9	10	11	12	13	**14**	15	16	17	18	19
D1 **D2**	-5 9994	-4 9995	-3 -3	-2 -2	-1 -1	0 0	1 1	2 2	3 3	4 -9995	5 -9994
Action	Protocol Error			Discard MSG			OK			Protocol Error	

P1 Routing Facility

A message routing facility is available at the P1 layer of MERVA Link USS. It routes an inbound conversation synchronously to another partner system.

Recipient Information in an Inbound Conversation

The first data that is received by a MERVA Link inbound conversation is a Probe PDU envelope with the data element identifier 0100. The Probe PDU envelope contains the recipient node name as part of the recipient address data element. The recipient node name must match the MERVA Link node name of the local (receiving) system. Inbound conversation data must not be delivered by the inbound P1 processor to a receiving application if the two node names do not match.

If the two node names do not match, a P1 processor can reject an inbound conversation and tell the sending process that the recipient node is not the MERVA Link node that actually received the data. This is what MERVA Link ESA does in the CICS and IMS environments.

As an alternative, a P1 processor can try to identify the partner system that houses the recipient MERVA Link node and route the inbound conversation to that partner system. This is what MERVA Link AIX and MERVA Link USS do.

P1 Routing Parameters

The MERVA Link USS customization parameters are divided into the following parameter groups:

- The unique local MERVA Link USS parameters
- The ASP parameter sets
- The ISC parameter sets

The ISC parameter sets are used by the P1 routing facility to route an inbound conversation to the appropriate destination. The ″invalid″ recipient node name is used as the key to find the parameters of the intersystem connection to the destination system. For more information about MERVA Link USS customization parameters, refer to the *MERVA for ESA Customization Guide*.

Each ISC parameter set is divided into two subsets:

- Parameters for routing a conversation via an SNA APPC connection
- Parameters for routing a conversation via a TCP/IP connection

Confirming a Routed Conversation

The confirmation of a conversation (that is, the confirmation of the correct transfer and delivery of a set of messages) is requested by a sending ASP. The request for confirmation can be part of a SUBMIT.Request that submits a message. It can also be submitted without a message.

The request for confirmation is received by the partner MTP either together with an inbound message or as separate control information.

A receiving MTP that delivers an inbound message to its ASP is informed in the DELIVER.Response whether the message was successfully delivered or not. It can therefore correctly respond to a request for confirmation at any time.

A receiving MTP that routes an inbound message to another node is informed of the successful delivery only if it requests a confirmation for the routed message. Therefore, it cannot correctly respond to a request for confirmation at any time unless it request a confirmation for every message. Requesting a confirmation for every message can decrease the message transfer rate significantly. This is why additional functionality is added to a receiving MTP to support message windowing in the gateway scenario.

Request for Confirmation (RFC) Parameter in a ProcessPDU.Indication

A request for confirmation (RFC) parameter is added to the parameters of a ProcessPDU.Indication service primitive. It is passed by an inbound TP (EKATPI or EKATCI) to the inbound P1 processor (EKAP1I). The inbound TP sets this parameter, if a request for confirmation has been received together with a message, when it passes a message to the P1 processor.

When the inbound TP that is not in an error state receives a request for confirmation as separate control information, it does not respond to the request for confirmation immediately. It calls the P1 processor with a specific ProcessPDU.Indication service primitive that contains the request for confirmation parameter in the service primitive parameter area. The pointer to the message envelope is NULL in that service primitive.

Handling a request for confirmation (RFC) in a ProcessPDU.Indication

The inbound MTP (P1 processor) handles a request for confirmation differently depending on whether it must deliver messages to its local ASP or route inbound messages to a partner node:

- When it must deliver inbound messages to its local ASP, it need not perform any special action as response to a request for confirmation that is not associated with a message. It returns immediately to the caller and indicates successful completion in the ProcessPDU.Response.
- When it must route inbound messages to a partner node, it must also pass the request for confirmation to the partner. If that request is not associated with a message, it issues a SendPDU.Request without application data to the applicable outbound TP (EKATPO or EKATCO).

Chapter 3. Peer-to-Peer Protocols

A peer-to-peer protocol is a convention between two functional entities at corresponding layers (peers) about their communication. The MERVA Link functional entities (Message Transfer and Application Support) communicate with their peer entities by exchanging data. The MERVA Link peer-to-peer protocols are therefore conventions about the syntax and the semantics of data exchanged between MERVA Link Message Transfer and MERVA Link Application Support.

The MERVA Link Command Transfer is another set of functions that use a proprietary peer-to-peer protocol. These functions support MERVA ESA and MERVA Link commands and their responses, rather than messages.

The unit of data exchanged by peer entities is called a protocol data unit (PDU).

MERVA Link Protocol Data Units (PDUs)

A protocol data unit (PDU) is a data construct exchanged between functionalities in corresponding layers of a message handling system, or more generally, of a communication system.

A PDU exchanged between two functionalities in a specific layer is assigned a name that reflects the purpose of this layer. For example, a PDU exchanged between transport layer functions is called a transport PDU (TPDU), and a PDU that is exchanged between message transfer layer functions is called a message PDU (MPDU).

A PDU exchanged between functionalities in a specific layer can contain a higher layer PDU as its content. For example, an MPDU can contain an Application Support PDU (ASPDU) as its content.

MERVA Link Message Handling System PDU Hierarchy

The PDUs exchanged between Message Transfer Functions in the MERVA Link message handling system are called Message PDUs (MPDU). The purpose of an MPDU qualifies it as either an Application Message PDU (AMPDU) or a Service Message PDU (SMPDU).

The PDUs exchanged between application support functions in the MERVA Link message handling system are called application support PDUs (ASPDUs). The purpose of an ASPDU qualifies it as either an interapplication messaging ASPDU (IM-ASPDU) or a status report ASPDU (SR-ASPDU).

The MERVA Link message handling system PDU hierarchy is shown in Figure 7 on page 30. A single line connecting two PDU boxes means "can have the format of", for example, an MPDU can have the format of an AMPDU or an SMPDU. A bold line connecting two PDU boxes means "contains", for example, an AMPDU contains an AMPDU Content.

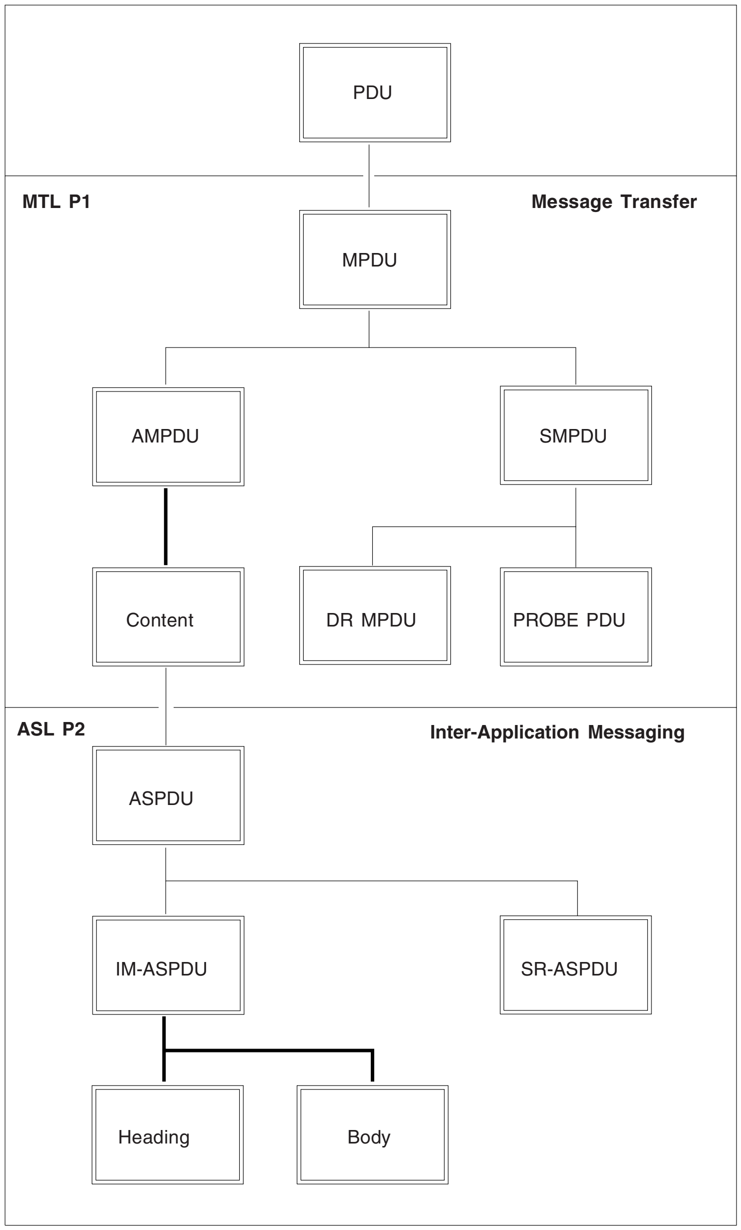

Figure 7. MERVA Link MHS PDU Hierarchy

MERVA Link MHS PDU Types

The MERVA Link P1 and P2 protocols are represented by protocol data units (PDUs). Four types of PDUs are defined in the MERVA Link message handling system:

- Probe PDU (an SMPDU)
- Delivery report MPDU (DR MPDU, also an SMPDU)
- AMPDU containing an application message (IM-ASPDU)
- AMPDU containing an acknowledgment message (SR-ASPDU)

A MERVA Link PDU is composed of PDU data elements. A PDU data element has the format of a 4-byte prefix (LLID), which is usually but not necessarily followed by data-element data. LL in the data-element prefix represents a 2-byte binary field that contains the length of the data element including the length of the LLID field, and ID represents a 2-byte identifier of the data element.

Data-element levels are assigned to PDU data elements. A data element not contained in another data element is called a level-1 data element. A data element contained in the data of a level-1 data element is called a level-2 data element. A data element contained in the data of a level-2 data element is called a level-3 data element. The identifier of level-1 data elements starts with X'01' (implicit data element) or X'81' (explicit data element).

The last data element of every PDU is the PDU trailer.

Probe PDU

A Probe PDU is used for control communication between message transfer entities. It is used by an MTP, for example, to request an ASP availability test from its partner MTP.

A Probe PDU consists of the:

- Probe Envelope (ID=0100)
- PDU Trailer (ID=81FF)

Delivery Report MPDU (DR MPDU)

A delivery report is used by an MTP to report successful or unsuccessful delivery of an AMPDU to the originator of the latter PDU. The MTP uses this means to report the delivery of a message only if it cannot report the delivery synchronously to its partner MTP.

The support of the asynchronous communication protocol based on LU 6.1 has been dropped in MERVA Link of MERVA ESA V4. MERVA Link of MERVA ESA V4 supports only synchronous communication protocols that do not use delivery reports. The support of the delivery report MPDU is therefore dropped in MERVA Link of MERVA ESA V4.

AMPDU containing an Application Message (IM-ASPDU)

The main purpose of the MERVA Link message handling system is to exchange messages or application data between applications. An AMPDU containing an IM-ASPDU conveys this application data. It is, therefore, the most important PDU in message handling system. The event associated with this PDU at the sending side is a message submission (service primitive SUBMIT.Request). The event associated with this PDU at the receiving side is a message delivery (service primitive DELIVER.Indication).

An AMPDU containing an application message consists of the following:

- AMPDU envelope (ID=0102)
- IM-ASPDU heading (ID=0120)
- IM-ASPDU body part header (ID=8121)
- One of the following:
  - IM-ASPDU body part data segment (ID=8122 or 8132)
  - IM-ASPDU body part encrypted data segment (ID=8123 or 8133)
  - IM-ASPDU body part compressed data segment (ID=8126 or 8136)
  - IM-ASPDU body part encrypted compressed data segment (ID=8127 or 8137)
- PDU trailer (ID=81FF)

The message text is contained in one or more IM-ASPDU body part data segments. This text can be:

- Plain text in EBCDIC encoding (ID=8122)
- Encrypted text in EBCDIC encoding (ID=8123)
- Compressed text in EBCDIC encoding (ID=8126)
- Encrypted compressed text in EBCDIC encoding (ID=8127)
- Plain text in ASCII encoding (ID=8132)
- Encrypted text in ASCII encoding (ID=8133)
- Compressed text in ASCII encoding (ID=8136)
- Encrypted compressed text in ASCII encoding (ID=8137)

Plain message text is first compressed and then encrypted if the IM-ASPDU body part data segment type is 8127 or 8137. To recover plain text from such a body part data segment, the transmitted text must first be decrypted and then decompressed.

AMPDU containing an Acknowledgment Message (SR-ASPDU)

The actual processing status of a message received from a partner application can be reported by the receiving application to the sending application in a status report. A status report is also called an acknowledgment message or a receipt report. The event associated with this PDU at the sending side is, again, a message submission (service primitive SUBMIT.Request). The event associated with this PDU at the receiving side is, again, a message delivery (service primitive DELIVER.Indication).

An AMPDU containing an acknowledgment message consists of the:

- AMPDU Envelope (ID=0102)
- Status Report ASPDU (ID=0112)
- PDU Trailer (ID=81FF)

MERVA Link PDU Data Elements

A MERVA Link PDU data element consists of a 4-byte LLID prefix that is usually, but not always, followed by data. The first 2 bytes (the LL part of the LLID) indicate the length, in bytes, of the data-element. Because this length always includes the length the prefix itself, the minimum length is X'0004' (for a data element that consists only of a prefix, with no data). Unless otherwise stated, data elements have a variable length.

The third and fourth bytes of the prefix (the ID part of the LLID) indicate the ID of the data-element. The data element IDs are listed in “Appendix A. PDU Data Elements” on page 163. The third byte indicates the group to which the data element belongs. The data-element groups are shown in the following table:

Group	Group Description or Data Elements in this Group
01 02 81 82	Implicit Level-1 Message Transfer Data Elements Implicit Level-1 Command Transfer Data Elements Explicit Level-1 Message Transfer Data Elements Explicit Level-1 Command Transfer Data Elements
10 11 92 93 14 15 95 96	Originator Address, Descriptor, Security Info Recipient Address and Descriptor Message Identifiers Date and Time Message Transfer Processor Trace Receipt Report, Error Report Status Codes, Status Data, Error Report Information Message Attachments, Control Information, Correlation Data
A0 A1 A2	Application Free Form Name Address Elements Message Transfer Process Names
B0	Control Indicators Consisting of 1 Character
C0	Request for MIP Reset, PDM Indicator
7F FF	Implicit Application Defined Data Element Explicit Application Defined Data Element, Do not Care DE

The data elements in the group B0 consist of the data-element prefix and 1 character as data. The data elements in the group C0 consist of the data-element prefix only.

The data contained in a data element can have the format of a sequence of data elements. All level-1 data elements, except the data elements of the message body and the PDU trailer, contain other data elements (level-2 data elements), which in turn can contain other data elements (level-3 data elements). Level-3 is the lowest level that a data element in a MERVA Link PDU can have. A data element that contains another data element is called an *implicit data element*. A data elements that does not contain another data element, or that contains no data at all, is called an *explicit data element*.

The meaning of the identifier part (bytes 3 and 4) of the prefix of a data element at level 2 or 3 might not be unique; its meaning might depend on the higher level data element that contains it. For example, a timestamp data element is always encoded in the same way, but its meaning depends on its context. The timestamp data element in an AMPDU envelope specifies the submit timestamp, while in a DR content it specifies the delivery timestamp.

Notation of the P1, P2, and P3 Protocol Definitions

The structure of each MPDU or CPDU is described in the following sections using a special notation. The elements of this notation are as follows:

::= This character string denotes a definition and can be read as *has a valid format of*. It is always preceded by the name of the PDU construct defined in this statement. This character string is always followed by a set operator (CHOICE, SEQ, SEQ OF, or SET) and a set of data elements enclosed in braces {}, or by another PDU construct.

CHOICE
This set operator means that a choice must be made between the elements of the following set. Only one element of the set is applicable at one time.

SEQ This set operator means that all elements of the following set must appear in the sequence as listed in the set. Specific elements of the set can be qualified as optional.

SEQ OF
This set operator means that the following set appears once or more than once.

SET This set operator means that all elements of the following set can appear once or more than once in the PDU construct to be defined. The elements of the set can appear in any sequence in the PDU construct to be defined. Unless otherwise stated, MERVA Link supports only the first occurrence of a specific data element in a set of data elements.

[xxxx] This character string with hexadecimal digits for xxxx specifies the identifier (ID) in the LLID prefix of a data element. In an encoded PDU it is always preceded by the 2-byte length field that contains the length of the data element including the 4 bytes of the data element LLID prefix.

If the data element identifier [xxxx] is immediately followed by a blank, that data element consists of the prefix only. A description of the meaning of that data element follows after the blank in this case.

An element in the set of data elements enclosed by braces that has no data-element identifier [xxxx] will be defined later in that section.

(an,..) This qualifier indicates that the data-element data consists of alphanumeric data and is not further expanded. The set of alphanumeric characters comprises the characters $, @, and #, the letters A to Z in uppercase, and the digits 0 to 9.

The alphanumeric data of a variable length data element can have trailing blanks. These blanks are insignificant unless otherwise stated.

(nc,..) This qualifier indicates that the data-element data consists of numeric characters and is not further expanded. The set of numeric characters comprises the digits 0 to 9.

(ps,..) This qualifier indicates that the data-element data consists of a printable string and is not further expanded. The set of printable characters comprises all printable characters of the EBCDIC character set.

(xc,..) This qualifier indicates that the data-element data consists of hexadecimal characters and is not further expanded. The set of hexadecimal characters comprises the uppercase letters A to F and the digits 0 to 9.

(xs,..) This qualifier indicates that the data-element data consists of a hexadecimal string and is not further expanded. A hexadecimal string can contain all codes from hexadecimal 00 to hexadecimal FF.

(..,vnn)
This qualifier indicates that the data-element has a variable length. The maximum length of the data-element data is identified by nn. The minimum data-element data length is 1.

(..,fnn) This qualifier indicates that the data element has a fixed length. The fixed length of the data-element data is identified by nn.

optional
This attribute indicates that the data element is optional. Optional means, it might or might not be part of the PDU independent of all other data elements in the PDU.

Any data element without the attribute optional is a mandatory data element. All mandatory data elements must be contained in a PDU.

Definition of the Message Transfer Protocol (P1)

Two classes of an MPDU are defined in P1:

```
MPDU ::= CHOICE {ApplicationMPDU, ServiceMPDU}
```

- Application MPDUs carry messages. One type of an application MPDU is defined. It carries a message from an originating ASP toward a recipient ASP.
- Service MPDUs carry information about messages. Two types of a service MPDU are currently defined:

  ```
  ServiceMPDU ::= CHOICE {DeliveryReportPDU,
                          ProbePDU          }
  ```

  – The delivery report PDU carries a delivery or a nondelivery report to an originating ASP. The support of a delivery report PDU has been dropped in MERVA Link of MERVA ESA V4.

  – The Probe PDU carries a request for information about the availability of an ASP to the MTP supporting this ASP.

Every MPDU is divided into the following parts:

- The envelope, which contains the information that the MTL needs to transfer the MPDU to its intended destination
- The content, which is the primary information the MPDU is intended to convey, but that can, however, be empty
- The trailer

PDU Trailer

Every MPDU is terminated by a PDU trailer. A PDU trailer consists of an LLID data-element prefix, optionally followed by one character as data element data. The PDU trailer data supports a conversation control communication in the MERVA Link TCP/IP environment that corresponds to the SNA APPC control communication.

The PDU trailer data element prefix is encoded as:

```
PDU_trailer_prefix ::= [000481FF] or [000581FF]
```

The PDU trailer data element data can be empty or one of the following numeric characters:

0	Standard PDU trailer without additional information (standard trailer)
1	PDU trailer requesting the termination of the conversation (termination trailer)
4	PDU trailer indicating an error (error trailer)
8	PDU trailer requesting a confirmation (confirmation request trailer)

PDU trailers with data are used only in the MERVA Link TCP/IP enviroment.

The PDU trailer is not shown in the following PDU definitions.

Application Message PDU (AMPDU)

An application MPDU carries a message to its recipient ASP. It has three parts: an envelope, content, and a trailer (the trailer is not shown below):

```
ApplicationMPDU ::= SEQ {[0102]AMPDUEnvelope, AMPDUContent}
```

The type of the AMPDU content is specified as an AMPDU envelope parameter. The only AMPDU content type defined is MERVA Link P2.

AMPDU Envelope

The AMPDU envelope (data-element identifier 0102) is defined as follows:

```
AMPDUEnvelope ::= SET {[1001]originator Address,
                       [1101]recipient Address,
                       [9200]ASLContentID (ps,v16) optional,
                       [9201]MPDUIdentifier (ps,v16),
                       [9301]SubmitTimeStamp (nc,f12),
                       [1403]MTPTrace optional,
                       [B000]EncodedInformationType (an,f1),
                       [B001]ReqDeliveryNotification (nc,f1),
                       [B002]Priority (an,f1),
                       [B003]ContentType P2 (nc,f1)          }
```

An AMPDU envelope can contain other, user-defined data elements. These data elements are ignored by MERVA Link. The maximum length of an envelope supported by MERVA Link is 512 bytes.

Originator Address and Recipient Address

The originator address and recipient address are sets of data elements that name and describe resources of the originator and recipient of a message. Address information is provided by the originating ASP and used by the message transfer system to route the MPDU to the recipient. It is finally used by the recipient ASP to pass it to the intended recipient application:

```
originator Address ::= Address
recipient Address  ::= Address
```

ASL Content ID

The ASL content ID is a message identifier that can be assigned to the message by the ASP. It is passed to the message transfer process as a SUBMIT.Request parameter. The ASL content ID is returned to the Application Support Process in a SUBMIT.Confirmation and in a delivery notification.

The data of this data element can have trailing blanks. These trailing blanks are insignificant.

MPDU Identifier

The MPDU identifier is a message identifier that is unique within the MT Node. It is assigned to the message by the message transfer process when the SUBMIT.Request is processed. The MPDU Identifier is returned to the ASP in a SUBMIT.Confirmation.

The MPDU identifier together with the originator address is unique within the MTS. It is the main reference of the message within the message transfer system. It is contained in a delivery notification referring to this message.

In MERVA Link the MPDU identifier is generated by the MTSP as the 16-byte character representation of the S/390 system clock.

The data of this data element can have trailing blanks when it was not generated by MERVA Link. These trailing blanks are insignificant.

Submit Timestamp

The submit timestamp contains the date and time when the

SUBMIT.Request was accepted by the MTS. Date and time are specified in the fixed format *YYMMDDHHMMSS* (year, month, day, hour, minute, second).

MTP Trace

The MTP trace can consist of two external MTP names, the name of a sending and a receiving MTP:

```
MTPTrace ::= SEQ {[A201]ExtSendMTPName (an,v8) optional,
                  [A202]ExtRecvMTPName (an,v8) optional }
```

The MTP trace information can be used by a receiving MTP to identify the sending MTP. An MTP trace data element must be contained in an envelope if an MTP name is required by the receiving MERVA Link implementation.

A MERVA Link receiving process may be independent of one or both MTP names. The complete MTP trace data element or either of the MTP names are optional in this case.

Encoded Information Type

The encoded information type of a message is the information that appears in its content. In an , it is the type (format) of the body.

This PDU element consists of the 4-byte LLID prefix, plus one character indicating the encoded information type:

Q	For MERVA ESA queue format
B	For the queue format used by MERVA USE & Branch workstation programs
N	For network (line) format

The encoded information type can be blank (X'40') if the envelope does not contain a body (it contains, for example, a status report).

The queue format used by MERVA USE & Branch workstation programs is not supported by MERVA Link of MERVA ESA. A message with a body of that format can, however, be handled by a MERVA Link USS Gateway.

If Network Format is indicated, the applicable network identifier can be found in MERVA Link P2 rather than in this protocol (MERVA Link P1).

Request for Delivery Notification

The delivery notification indicator tells whether a delivery notification should be returned to the originating ASP. A nondelivery notification is always returned to the originating ASP.

This PDU element consists of the 4-byte LLID prefix and 1 character identifying the request for delivery notification as follows:

0	No delivery notification requested
2	Delivery notification requested

As the MERVA Link Message Integrity Protocol is a mandatory service element in MERVA Link, a delivery notification is always requested and returned.

Priority

The priority indicator specifies the relative priority of the message carried by the MPDU.

This PDU element consists of the 4-byte LLID prefix and 1 character identifying the priority as follows:

L	Low priority
N	Normal priority
H	High priority

A sending MERVA Link ASP sets the P1 priority according to the position of the applicable send queue in the MERVA Link send queue cluster. Messages from the first, second, and third send queue are assigned high, normal, and low priority, respectively.

Content Type

The content type specifies the encoding rules of the MPDU content. The only content type defined is MERVA Link P2 (content type data is 2).

Address

The address in an AMPDU envelope is as follows:

```
Address     ::= SET {[A100]MERVA_System_Type (an,f4) optional,
                     [A101]MTNodeName (an,v8),
                     [A102]ASProcessName (an,v8)             }
```

MERVA System Type

The MERVA System Type identifies the type and the version of the MERVA installation that houses the subject ASP. It consists of an alphabetic MERVA System Identifier, followed by three numeric characters (VRM) identifying the MERVA version. The alphabetic MERVA System Identifiers are defined as follows:

C	MERVA ESA CICS MVS
V	MERVA ESA CICS VSE
I	MERVA ESA IMS
U	MERVA USS
A	MERVA AIX
O	MERVA OS/2®
N	MERVA NT

MT Node Name

The MT node name identifies a MERVA Link system. This name must be unique within the message transfer system. It is the vehicle to address another MT Node in the MTS.

AS Process Name

The AS Process name identifies a specific application (ASP) within the MT Node. This name must be unique within an MT Node.

AMPDU Content

The AMPDU content contains the information to be communicated between the originating and receiving applications.

The format of the AMPDU content is described in “Definition of the Application Support Protocol (P2)” on page 41.

Delivery Report MPDU (DR MPDU)

MERVA Link of MERVA ESA V4 supports only synchronous communication protocols that do not use delivery reports. The support of the delivery report MPDU is therefore dropped in MERVA Link of MERVA ESA V4.

Probe PDU

A Probe PDU carries a request for information or a request for specific action to a partner system. A Probe PDU has only an envelope and no content.

```
ProbePDU ::= [0100]ProbeEnvelope
```

The Probe envelope contains all information the message transfer system requires to pass the PDU to the intended recipient, and to perform the requested function.

The Probe is supported by all MTPs that have a synchronous conversation with their partner MTP (SNA APPC, TCP/IP, or BTB). The Probe processing result is reported back to the sender of the Probe within the same conversation.

Probe Envelope

The Probe envelope (data-element identifier 0100) is defined as follows:

```
ProbeEnvelope  ::= SET {[1001]originator Address,
                        [1101]recipient Address,
                        [1403]MTPTrace optional,
                        [1003]ClientSecInfo optional,
                        [1004]ChangeSecInfo optional,
                        [B004]ProbeFunction (an,f1)  }
```

Address

The originator address parameter identifies the originator of the Probe. It can be a sending ASP that must test a partner ASP for its availability. As an alternative, it can be the name of a MERVA Link internal resource that requests a specific action from its partner resource in a partner MERVA Link node (for example, change client security information).

The recipient address parameter identifies the target application. It can be a receiving ASP that must be tested for its availability. As an alternative, it can be the name of a MERVA Link internal resource that performs a requested action (for example, change client security information).

MTP Trace

The MTP trace contains a sending and a receiving MTP Name. It can be used for partner verification at the receiving side. The MTP trace data element need not to be sent to partner systems that use another method to identify the recipient MTP and to verify the originator.

Client Security Information

Client security information (data-element identifier 1003) is defined as follows:

```
ClientSecInfo  ::=    SecurityInfo
```

Client security information conveys data that enables the receiving MTP to check whether the sender (client) is authorized to access the receiving MERVA Link system (server). Partner authorization is performed only when the DC system used by MERVA Link does not support conversation security (for example, TCP/IP). Client security information is mandatory in that case. Otherwise it is optional data that is ignored by the MERVA Link server.

Change Security Information

Change security information (data-element identifier 1004) is defined as follows:

```
ChangeSecInfo  ::=    SecurityInfo
```

Change security information conveys data that enables the receiving MTP to change the conversation security information that must be used by its associated sending MTP to establish a conversation with its partner MTP. Change security information is mandatory if the Probe function is C (identifying a change security information request). Otherwise it is optional data that is ignored by the MERVA Link server.

Probe Function

Probe Function indicates the purpose of this Probe PDU. This PDU element consists of the 4-byte LLID prefix and one character identifying the purpose of the Probe as follows:

T	**Test** the availability of a receiving ASP.
R	Test the availability of a receiving ASP and ask the receiving ASP to **return** a probe (with function T).
C	Request the **change** of client security information in the partner MERVA Link node.
K	Test the availability of a receiving ASP and **kick off** the corresponding sending ASP at the end of the receiving process.
S	Test the availability of a receiving ASP and **start** the corresponding sending ASP at the end of the receiving process.
H	Test the availability of a receiving ASP and **hold** the corresponding sending ASP at the begin of the receiving process.
E	**Enable** the receiving ASP and test its availability.
D	Test the availability of a receiving ASP and **disable** it at the end of the receiving process.

Any MERVA Link implementation must support the T-Probe. The support of all other Probe functions is optional.

Security Information

Security information in a Probe envelope is defined as follows:

```
SecurityInfo   ::= SET {[A108]UserID (an,v8),
                        [A109]EncrPasswd (xs,f8),
                        [A10A]EncrCtlInfo (xs,v32),
                        [B005]EncrMethod (an,f1)   }
```

All security information data elements are mandatory.

User ID

The User ID data element contains the applicable security user identifier.

Encrypted Password

The encrypted password is the password that is currently applicable for the specified security user. The password has been encypted for confidentiality reasons during the transfer. It must be decrypted as specified by the encryption control information and the encryption method before it can be used.

Encryption Control Information
The encryption control information and the encryption method enable a MERVA Link function to decrypt the encrypted password.

Encryption Method
The encryption method identifies the function that was used by MERVA Link to encrypt the password as follows:

0	MERVA Link basic password encryption method
1	MERVA Link AIX Crypt
2	MERVA Link USS Crypt
3	MERVA Link NT Crypt

The basic password encryption method is supported by all MERVA Link implementations that support TCP/IP. The encryption method that is based on the unrestricted DES function **crypt()** can be used only for connections between MERVA Link nodes in the same operating system environment. This method can, for example, not be used in a connection from MERVA Link AIX to MERVA Link USS.

Both the basic password encryption method and the password encryption method based on function crypt() use MERVA Link proprietary algorithms. These algorithms are confidential, and are disclosed on a need-to-know basis only.

Definition of the Application Support Protocol (P2)

Two types of an ASPDU are defined in P2:
- IM-ASPDUs, which carry interapplication messages
- SR-ASPDUs, which carry interapplication messaging status reports

```
ASPDU ::= CHOICE {IM-ASPDU, SR-ASPDU}
```

Interapplication Message ASPDU (IM-ASPDU)

Every IM-ASPDU has two parts, the heading and the body.

```
IM-ASPDU ::= SEQ {[0120]Heading, Body}
```

IM-ASPDU Heading

The heading of an IM-ASPDU contains descriptive information or control data concerning the application message:

```
Heading ::= SET {[1002]originator ApplicationDescriptor,
                 [1102]recipient ApplicationDescriptor,
                 [9202]IAMessageID (ps,v16),
                 [9203]MIPMessageID (xs,f8) optional,
                 [9204]MIPMessageSN (nc,f4),
                 [9600]BodyPartEncryptionCtrlInfo (xs,v60) optional,
                 [9601]MsgAuthCtrlInfo (xs,v60) optional,
                 [9602]MERVALineFormatID (an,f1) optional,
                 [9603]MERVAMessageType (an,v8) optional,
                 [9604]MIPWindowSize (nc,f3),
                 [9605]BodyPartCompressionCtrlInfo (xs,v60) optional,
                 [9608]Buckslip (ps,v256) optional,
                 [9609]Subject (ps,v60) optional,
                 [9610]ApplRequestData (ps,v1024) optional,
                 [9611]ApplResponseData (ps,v256) optional,
                 [9612]ApplAckData (ps,v256) optional,
                 [9613]ApplMacData (ps,v256) optional,
                 [9614]ApplPacData (ps,v256) optional,
                 [B000]BodyType (an,f1),
                 [B001]ReqReceiptConfirmation (nc,f1),
                 [B002]Priority (an,f1),
                 [C000] MIPResetIndicator optional,
                 [C001] PDMIndicator optional                      }
```

The MERVA ESA line-format identifier is mandatory if the body type denotes a MERVA ESA line format. The MERVA ESA message-type data element is optional regardless of the body type.

A message heading can contain user-defined data elements. These data elements are ignored by MERVA Link. MERVA Link supports a maximum message heading length of 4084 bytes. The maximum length of the message heading and the appended body part header is 4096 bytes (4KB).

Application Descriptor

The application descriptor is defined as follows:

```
ApplicationDescriptor ::= SET {[A001]FreeFormName (ps,v60) optional,
                               [A101]MTNodeName (an,v8),
                               [A102]ASProcessName (an,v8)         }
```

The free-form name identifies the application in a format selected for human comprehension rather than for processing by computer processes.

The MT node name identifies the message transfer function in a MERVA Link system. This name must be unique within the message transfer system. It is the vehicle to address another MT Node in the MTS and to verify the identity of a communication partner.

The AS Process Name identifies a specific application within the MT Node. This name must be unique within an MT Node.

IAM Message ID

The Interapplication Messaging message identifier identifies a particular message. It is used to provide the MERVA Link P2 Message Identification service element. The IAM message ID is intended to be a unique and unambiguous identifier within the originating application’s context.

MIP Message Identifier

The Message Integrity Protocol (MIP) message identifier component conveys a message identifier generated by the sending application support process. It is used to provide the Message Integrity Protocol service element.

MIP Message Sequence Number

The Message Integrity Protocol Sequence Number component conveys a message sequence number generated by the sending application support process. It is used to provide the Message Integrity Protocol service element.

Body Part Encryption Control Information

The Encryption component conveys the indication that the message body is encrypted and, optionally, either the name of an encryption algorithm or an encryption key, or both. As a matter of fact, the meaning of the data-element data must be bilaterally agreed upon between the cooperating ASPs. The body part data segment identifier indicates whether the body is actually encrypted (8123, 8127, 8133, or 8137) or in plain text (8122, 8126, 8132, or 8136).

Body part text that must be compressed and encrypted must first be compressed and then encrypted. Body part text that is both compressed and encrypted must first be decrypted and then decompressed (expanded).

Message Authentication Control Information
The authentication component conveys the indication that the message body was authenticated and the result of the authentication. The authentication algorithm or its name can also be part of this data element. The meaning of the data-element data must be bilaterally agreed upon between the cooperating ASPs.

MERVA ESA Line-Format Identifier
The MERVA ESA line-format identifier component conveys the line-format identifier used when the message was formatted from MERVA ESA TOF format to the MERVA ESA line (net) format. This parameter is applicable only if the body type is MERVA ESA network format (N). It is a mandatory data element then.

MERVA ESA Message Type

The MERVA ESA message-type component conveys the type of the message as specified when the message was generated in MERVA ESA. It is used by MERVA ESA Message Format Services to map the message via the MERVA ESA TOF format to any other format defined in the MCB for that message type.

This parameter is applicable only if the body type is MERVA ESA Network Format (N). If this data element is missing, the receiving process should determine the message type from the data in the ASPDU body.

The data of this data element can have trailing blanks. These trailing blanks are insignificant.

MIP Window Size
The MIP window size component conveys the message transfer window size used to provide the MIP service element.

Body Part Compression Control Information
The compression component conveys the indication that the message body is compressed and, optionally, either the name of a compression/expansion algorithm or a compression/expansion key, or both. As a matter of fact, the meaning of the data element data must be bilaterally agreed upon between the cooperating ASPs. The body part data segment identifier tells whether the body part text is actually compressed (8126, 8127, 8136, or 8137) or not compressed (8122, 8123, 8132, or 8133).

Body part text that must be compressed and encrypted must first be compressed and then encrypted. Body part text that is both compressed and encrypted must first be decrypted and then decompressed (expanded).

Buckslip

The Buckslip component conveys a short notice attached to the subject message.

Subject
The subject component conveys information the originator specified as the subject of the message.

Application Request Data

The Application Request Data component conveys application specific request data.

Application Response Data

The Application Response Data component conveys application specific response data.

Application Acknowledgment Data

The Application Acknowledgment Data component conveys application specific acknowledgment data.

Application MAC Data

The Application MAC Data component conveys application specific Message Authentication Code information.

Application PAC Data

The Application PAC Data component conveys application specific PAC information.

Body Type

The body-type indicator specifies the type of any body part contained in the body. The data of this data element consists of one character indicating the body type:

Q	For MERVA ESA queue format
B	For the queue buffer format of a MERVA USE & Branch workstation program
N	For network (line) format

The queue format used by MERVA USE & Branch workstation programs is not supported by MERVA Link of MERVA ESA.

Request for Receipt Confirmation

The Receipt Confirmation indicator tells whether either a receipt or a nonreceipt confirmation should be returned to the originating ASP. The receipt confirmation implies the nonreceipt notification. The data of this data element consists of 1 character identifying the requested notification. Its meaning is as follows:

0	No receipt confirmation requested
1	Nonreceipt notification requested
2	Receipt confirmation requested.

Priority

The priority parameter specifies the relative priority of the message carried by the IM-ASPDU. The data of this data element consists of 1 character identifying the priority as follows:

L	Low priority
N	Normal priority
H	High priority.

MIP Reset Indicator

The MIP reset indicator conveys the information that the specified IAM Message Sequence Number must be accepted as a Message Integrity Protocol syncpoint. A Message Integrity Protocol violation cannot be reported for a message that contains this indicator. This PDU element consists of the 4-byte LLID prefix only.

PDM Indicator

The PDM indicator indicates that the message might have already been passed to the receiving application via a route outside the scope of MERVA Link, for example, TELEX. A message containing this indicator should be checked by a process outside the scope of MERVA Link to prevent duplication. This PDU element consists of the 4-byte LLID prefix only.

This data element is sent to a partner ASP if corresponding information is found in the message (MERVA Link control field EKAPDUPM with the content PDM). The receiving MERVA Link ASP provides the PDM indicator in the message (characters PDM in the MERVA Link control field EKAPDUPM).

IM-ASPDU Body

The body of an IM-ASPDU consists of a single body part. The type of that body part is specified in the BodyPartType data element in the message heading.

```
Body    ::= BodyPart
```

The body part is defined as follows:

```
BodyPart ::= SEQ {[8121]BodyPartHeader (xs,f8),
                 SEQ OF CHOICE {[8122]BPDataSeg (xs,v32763),
                                [8123]EncrBPDataSeg (xs,v32763),
                                [8126]ComprBPDataSeg (xs,v32763),
                                [8127]EncrComprBPDataSeg (xs,v32763),
                                [8132]BPDataSeg (xs,v32763),
                                [8133]EncrBPDataSeg (xs,v32763),
                                [8136]ComprBPDataSeg (xs,v32763),
                                [8137]EncrComprBPDataSeg (xs,v32763) }}
```

Body Part Header

The body part header defines the begin of a body part. It has two 4-byte fields as data element data. The first 4-byte field contains the total data length of this body part as a 4-byte binary number. The second 4-byte field contains the total data length of the body. The content of these two fields is identical as only one body part is supported in a body.

Body Part Data Segments

The body part header is followed in the body part by a sequence of body part data segments. All data segments of a body part have the same identifier. The segmentation of the body part data has no meaning for that data. It is not correlated to an internal structure of the body part data.

Body part data can be divided into any number of segments. The length of a body part data segment (including the 8-byte data segment prefix) can be in the range of 9 bytes to 32K minus 1 bytes.

A body part data segment can be encrypted. Whether it is actually encrypted or not is indicated in the data element identifier of the body part data segment. ID=8122 and ID=8132 identify a segment with plain text, ID=8123 and ID=8133 identify an encrypted body part data segment, ID=8126 and ID=8136 identify a segment with compressed plain text, ID=8127 and ID=8137 identify an encrypted compressed body part data segment.

Encryption control information, which enables the receiver of an encrypted segment to recover the plain text, can be contained in the message heading.

Compression control information, which enables the receiver of a compressed segment to expand the text, can be contained in the message heading.

The body part data segment data starts with a 4-byte field that can contain the data length of this body part data segment (data element length minus 8). Body part data follow this 4-byte field. The minimum length of a body part data segment is therefore 9 bytes.

The encoding of body part data segment data is EBCDIC for ID=8122, ID=8123, ID=8126, and ID=8127. For ID=8132, ID=8133, ID=8136, and ID=8137 the encoding is ASCII. EBCDIC and ASCII body part data segments are handled by MERVA Link of MERVA ESA in the same way. If code conversion is necessary it must be performed in an application support filter.

Status Report ASPDU (SR-ASPDU)

The SR-ASPDU is used to return notification of receipt or nonreceipt of a message to its originator:

```
SR-ASPDU ::= [0112]ReceiptInfo
```

The receipt information contains receipt or nonreceipt information, information to correlate the report to the applicable message and the recipient, and MERVA Link Message Integrity Protocol control information.

A status report can contain user-defined data elements. These data elements are ignored by MERVA Link. MERVA Link supports a maximum status report length of 4084 bytes:

```
ReceiptInfo ::= SET {[1102]reported RecApplDescriptor optional,
                     [9202]reported IAMessageID (ps,v16),
                     [9203]MIPMessageID (xs,f8) optional,
                     [9204]MIPMessageSN (nc,f4),
                     [9604]MIPWindowSize (nc,f3),
                     [9608]Buckslip (ps,v256) optional,
                     [9609]Subject (ps,v60) optional,
                     [9611]ApplResponseData (ps,v256) optional,
                     [9612]ApplAckData (ps,v256) optional,
                     [9613]ApplMacData (ps,v256) optional,
                     [9614]ApplPacData (ps,v256) optional,
                     [C000]MIPResetIndicator optional,
                     [1500]Report                               }
```

Recipient Application Descriptor
The Recipient Application Descriptor specifies the recipient of the reported message. It describes the originator of the status report.

IAM Message ID
The IAM message ID identifies the reported message.

Standard P2 Data Elements
A subset of the MERVA Link P2 data elements that are defined for an IM-ASPDU heading are also defined for an SR-ASPDU. This subset consists of the following data elements:

- 9203 MIP Message ID
- 9204 MIP Message Sequence Number
- 9604 MIP Window Size
- 9608 Buckslip
- 9609 Subject

- 9611 Application Response Data
- 9612 Application ACK Data
- 9613 Application MAC Data
- 9614 Application PAC Data
- C000 MIP Reset Indicator

For more information about these data elements, refer to the description of the IM-ASPDU heading.

Report

The report data element contains the main information conveyed in a status report. It can, however, also be used by a receiving MERVA Link process to return error information in case of a receiving process failure (conversation error report). Specific rules for the layout and content of a report data element may apply in the latter case.

The report data element contains the following information:

```
Report      ::= SET {[9301]ReportTime optional,
                     [9501]ReturnCode (nc,f2),
                     [9502]DiagnosticCode (ps,f6) optional,
                     SEQ OF {[9503]ReportData (xs,v256) optional},
                     [B006]DcOriginatorType (an,f1) optional,
                     [9505]IntErrorCodeVector (xs,f12) optional,
                     SEQ OF {[9506]OperatorMsg (ps,v79) optional} }
```

The maximum length of a Report data element in a Status Report is 4080 bytes. The maximum length of a Report data element that is used by a receiving TP to report an error synchronously is 1024 bytes.

Report Time

The report time specifies the date and time when the message was received by the recipient application or the date and time when the report was generated. Its format is *YYMMDDHHMMSS* (year, month, day, hour, minute, second). The report time is missing from a conversation error report.

Return Code

The return code specifies whether the notification is for receipt or nonreceipt. Receipt can be qualified as intermediate or final receipt.

```
ReturnCode ::= CHOICE {00 final receipt,
                       04 not-final (intermediate) receipt,
                       08 final nonreceipt                }
```

Final receipt means that message processing in the receiving application is finished and no other status report will be sent. Not-final receipt means, that message processing in the receiving application is not yet finished and another status report will be sent. A nonreceipt is always treated as a final status report.

The return code in a conversation error report can be 04 (warning), 08 (error), or 12 (severe error).

Diagnostic Code

The diagnostic code data element provides additional information about the reported status. It can be a mnemonic or a sequence of MERVA Link error codes as specified by the originator of the diagnostic code.

Diagnostic code values are not architected by MERVA Link. Each MERVA Link implementation defines its own set of diagnostic codes. This is why the type of the MERVA Link system that generated a diagnostic code must be known in order to interpret a diagnostic code.

Report Data
You can arrange for additional receipt report information to be contained in a sequence of report-data data elements. The meaning of report data is defined by an application using MERVA Link services rather than by the MERVA Link architecture. Corresponding functionality can be implemented in a MERVA Link user exit as application communication functionality (ACF).

Diagnostic Code Originator Type
The diagnostic code originator type data element contains information about the type of the MERVA Link system that generated the diagnostic code in this report. The diagnostic code originator types are defined as follows:

E	MERVA ESA (CICS MVS, CICS VSE, or IMS)
U	MERVA USS
A	MERVA AIX
O	MERVA OS/2
N	MERVA NT

MERVA Link Internal Error Code Vector (ECV)
Any MERVA Link internal error information is contained in a report data element in the format of a fixed length sequence of six two-byte error codes called an error code vector (ECV). The meaning of the error codes is specified by the MERVA Link implementation identified by the diagnostic code originator type data element.

Operator Message
An explanation of the return code, the diagnostic code, and the internal error code vector can be contained in a sequence of operator messages.

Definition of the Command Transfer Protocol (P3)

A command transfer service is defined for use by the MERVA System Control Facility. It employs a pair of command transfer processors (CTPs): an outbound CTP and an inbound CTP. The outbound CTP is also called the requestor that sends a command. The inbound CTP is also called the server that receives a command and sends the command response back to the requestor.

The rules for the communication between the two CTPs are defined in the MERVA Link Command Transfer Protocol. This protocol is referred to as P3. It defines the syntax and the semantic of the data exchanged between the CTPs, called Command Protocol Data Units (CPDUs), and the characteristics of the PDU exchange using APPC (Advanced Program-to-Program Communication).

A CPDU exchanged between Command Transfer Processors in the MERVA System Control Facility has two parts, an envelope and a content:

```
CPDU ::= SEQ {[0202]CPDUEnvelope, CPDUContent}
```

The CPDU envelope contains information to route the command to its intended command processor or to route the response back to the command requestor. The CPDU content contains the command request or the command response.

Command Transfer PDU Envelope (CPDUEnvelope)

The CPDU envelope (data-element identifier 0202) is defined as follows:

```
CPDUEnvelope ::= SET {[1001]originator Address,
                      [1101]recipient Address,
                      [B000]EncodedInformationType (an,f1) }
```

A CPDU envelope can contain other, user-defined data elements. These data elements are ignored by MERVA Link. The maximum length of a command PDU envelope supported by MERVA Link is 256 bytes.

Address

The originator and the recipient address is defined as a set of data elements describing and naming resources of the originator or the recipient of a CPDU.

```
Address     ::= SET {[A101]Node Name (an,v8),
                     [A102]UserName (an,v8)  }
```

The node name identifies a MERVA Link system in a network of MERVA systems. This name must be unique within this network.

The user name identifies a specific user or a command processor within that node.

Encoded Information Type

The encoded information type of a message is the information that appears in its content. In a CPDU it is the type (format) of the body.

This PDU element consists of the 4-byte LLID prefix and 1 character identifying the encoded information type as follows:

Q	Command response in queue format.
N	Command response in network (line) format.

The encoded information type is blank (X'40') if the content does not contain a body.

Command Transfer PDU Content (CPDUContent)

The purpose of a CPDU Content qualifies it as either a Command Request PDU (CRqPDU) or a Command Response PDU (CRsPDU).

```
CPDUContent ::= CHOICE {CRqPDU, CRsPDU}
```

A Command Reqest PDU carries a MERVA operator command and other control information from a command requestor to a command server. A Command Response PDU carries the response of the command server back to the command requestor.

Command Request PDU (CRqPDU)

A Command Request PDU carries a command to the command processor in the target system. The CRqPDU consists of a single part, the Command Request Heading.

```
CRqPDU ::= [0220]CRqHd
```

Command Request Heading

The Command Request Heading contains the command and descriptive information or control data concerning the command.

```
CRqHd   ::= SET {[1002]originator Descriptor,
                 [1102]recipient Descriptor,
                 [9603]MERVAMessageType (an,v8),
                 [9608]Buckslip (ps,v256) optional,
                 [960A]UniqueMscCorData (ps,v64) optional,
                 [960B]SpecMscCorData (ps,v256) optional,
                 [960C]SpecCmdCorData (ps,v1024) optional, [B000]BodyType (an,f1)
```

A command request heading can contain user-defined data elements. These data elements are ignored by MERVA Link. MERVA Link supports a maximum command request heading length of 2048 bytes.

Descriptor

The originator and recipient descriptors are defined as follows:

```
Descriptor ::= SET {[A001]FreeFormName (ps,v60) optional,
                    [A101]NodeName (an,v8),
                    [A102]UserName (an,v8)               }
```

The free-form name identifies the user or application in a format selected for human comprehension rather than for processing by computer processes.

The node name identifies the MERVA Link system in a network of MERVA systems. This name must be unique within this network.

The User Name identifies a specific user or application within that node.

Message Type

The message-type component conveys the type of the screen in which the command was entered.

The data of this data element can have trailing blanks. These trailing blanks are insignificant.

Buckslip

The Buckslip data element contains the command as it was entered by the operator.

Unique MSC Correlation Data

The unique correlation data is provided by a command processor and describes its status when a command has been executed. Unique MSC correlation data is optional in a command request. It must be returned to the command processor in the next command request if the execution of that command depends on the result of the previous command.

The semantic of the unique MSC correlation data is not defined in P3. It must be bilaterally agreed between the cooperating System Control Processes (SCPs).

Specific MSC Correlation Data

Specific MSC correlation data is provided by a lower level command execution routine and describes its status when a command has been executed. It must be returned to that routine in the next command request if the execution of that command depends on the result of the previous command.

The semantic of the specific MSC correlation data is not defined in P3. It is defined by the command execution routine.

Specific CMD Correlation Data

Specific command correlation data contains the command response in a specific command sequence, for example, the DDS command sequence. It must be passed to the command execution routine like a parameter each time the command is executed.

The semantic of the specific command correlation data is not defined in P3. It is defined by the command execution routine of that specific command.

Body Type

The body-type indicator specifies the type of the body. It is blank to indicate that no body follows this command request heading.

Command Response PDU (CRsPDU)

A Command Response PDU carries a command response from the partner system back to the requestor. A CRsPDU can have two parts, the heading and the body. The heading is mandatory, the body is optional.

```
CRsPDU   ::= SEQ {[0221]CRsHd,
                  [8221]CRsBody optional }
```

Command Response Heading

The heading of a CRsPDU contains descriptive information or control data concerning the command response.

```
CRsHd   ::= SET {[1002]originator Descriptor,
                 [1102]recipient Descriptor,
                 [1500]Report,
                 [9603]MERVAMessageType (an,v8) optional,
                 [9608]Buckslip (ps,v256) optional,
                 [960A]UniqueMscCorData (ps,v64) optional,
                 [960B]SpecMscCorData (ps,v256) optional,
                 [B000]BodyType (an,f1)                   }
```

A command response heading can contain user-defined data elements. These data elements are ignored by MERVA Link. MERVA Link supports a maximum command response heading length of 1024 bytes.

Descriptor

The originator and recipient descriptors are defined as follows:

```
Descriptor ::= SET {[A001]FreeFormName (ps,v60) optional,
                    [A101]NodeName (an,v8),
                    [A102]UserName (an,v8)               }
```

The free-form name identifies the user or application in a format selected for human comprehension rather than for processing by computer processes.

The node name identifies the MERVA Link system in a network of MERVA systems. This name must be unique within this network.

The User Name identifies a specific user or application within that node.

Report

The information contained in the report data element provides the timestamp of the partner system and tells whether command processing was successful or unsuccessful:

```
Report      ::= SET {[9301]ReportTime (nc,f12),
                     [9501]ReturnCode (nc,f2),
                     [9504]CpDiagCode (ps,f8) optional }
```

- The report time specifies the date and time when the report was generated. Its format is *YYMMDDHHMMSS* (year, month, day, hour, minute, second).
- The return code tells whether the command was successfully processed or not successfully processed.

  ```
  ReturnCode ::= CHOICE {00 command processing successful,
                         08 command processing error        }
  ```
- Information about the nature of a command processing error might be provided by the command processing diagnostic code. Whether the diagnostic code is set depends upon the implementation and is not defined in P3.

Message Type

The message-type component conveys the type of the message. It is used by MERVA Message Format Services to map the message to TOF format for display at the user's terminal. This parameter is applicable only if the body type is N.

The data of this data element can have trailing blanks. These trailing blanks are insignificant.

Buckslip

The Buckslip component conveys a short notice attached to the command response. It can, for example, be displayed in the error message line of an EUD screen.

Unique MSC Correlation Data

The unique correlation data is provided by a command processor and describes its status when a command has been executed. Unique MSC correlation data is optional in a command request. It must be returned to the command processor in the next command request if the execution of that command depends on the result of the previous command.

The semantic of the unique MSC correlation data is not defined in P3. It must be bilaterally agreed between the cooperating System Control Processes (SCPs).

Specific MSC Correlation Data

Specific MSC correlation data is provided by a lower-level command execution routine and describes its status when a command has been executed. It must be returned to that routine in the next command request if the execution of that command depends on the result of the previous command.

The semantic of the specific MSC correlation data is not defined in P3. It is defined by the command execution routine.

Body Type

The body-type indicator specifies the type of the body. It is N if the body contains the command response in the MERVA net format. It is Q if the body contains the command response in the MERVA queue format (compressed TOF format). The body type is blank if the body is missing.

Command Response Body

The body of a CRsPDU consists of a single body part. It contains the command response data in MERVA net or queue format. The message type of a command response in net format is contained in the Message Type data element of the CRsPDU heading. The line format identifier is the first character of the message type.

The maximum length of a command response body is 6908 bytes.

Command PDU Trailer

Every CPDU is terminated by a Command PDU trailer. A CPDU trailer consists of an LLID data-element prefix only. It has no data and is encoded as

```
CPDUtrailer ::= [000482FF]
```

Command Error Report

The receiver of a Command Request PDU can return an error report instead of a command response PDU if it finds an error during command request processing. The error report is defined as follows:

```
ErrorReport ::= [1500]Report

Report      ::= SEQ {[9301]ReportTime (nc,f12),
                     [9501]ReturnCode (nc,f2),
                     [9504]CpDiagCode (ps,f8)  }
```

The Report Time specifies the date and time when the report was generated. Its format is *YYMMDDHHMMSS.* The Return Code is 08. Information about the nature of a command processing error or the incomplete command processing is contained in the Command Processing Diagnostic Code. The semantic of the diagnostic code is not defined in P3.

All three data elements of the Report data element are mandatory in an error report.

Chapter 4. Boundary Function Service Primitives

The Layered Representation of the MERVA Link message handling system model comprises two layers, the application support layer and the message transfer layer. The latter layer is divided into two sublayers, the upper sublayer that contains the message transfer service processor, and the lower sublayer that contains the message transfer processors.

These three sublayers constitute four boundaries. These are the upper sublayer boundaries of the three sublayers, and the lower sublayer boundary of the lowest sublayer. Figure 8 shows the three MERVA Link sublayers and the four boundaries.

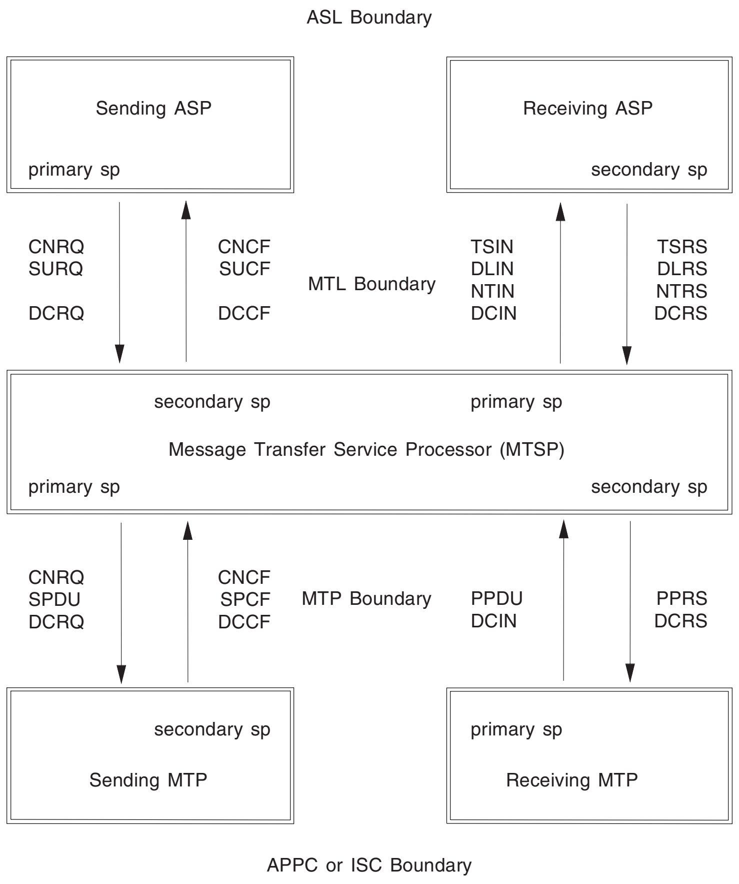

Figure 8. MERVA Link Sublayer Boundaries and Service Primitives

The main layer boundary within the MERVA Link message handling system is the boundary between the ASL and the MTL, called the MTL boundary.

The MTL internal boundary between the message transfer service processor and a message transfer processor is also described by service primitives. This boundary is called the message transfer processor boundary.

MTL Boundary Function Service Primitives

An application support processor communicates with the message transfer service processor or an application support filter using the following set of service primitives:

CONNECT.Request/Confirmation (CNRQ/CNCF)
A CONNECT.Request is issued by an ASP to establish a conversation with its partner ASP for the transfer of a set of messages.

A CONNECT.Confirmation is the response of the MTL to a CONNECT.Request. A CONNECT.Confirmation indicates whether a connection to the partner was acquired and if the partner ASP is prepared to receive messages.

TEST.Indication/Response (TSIN/TSRS)
A TEST.Indication is issued by the MERVA Link message transfer layer to test for the availability of a receiving ASP. This service primitive corresponds to a CONNECT.Request that includes the request to test the availability of the receiving ASP.

A TEST.Response is issued by the receiving ASP to indicate whether it is prepared to accept messages.

SUBMIT.Request/Confirmation (SURQ/SUCF)
A SUBMIT.Request is issued by a sending ASP to request the transfer of a message or the transfer of a status report.

A SUBMIT.Confirmation is the response of the MTL to a SUBMIT.Request. A SUBMIT.Confirmation indicates whether the MERVA Link message transfer system has accepted the request to transfer the data in the SUBMIT.Request to the recipient application, or whether an error has been found and the request was not accepted. A SUBMIT.Confirmation can even indicate that the message was successfully delivered to the recipient ASP and the ASP has taken responsibility of that message.

DELIVER.Indication/Response (DLIN/DLRS)
A DELIVER.Indication delivers a message or a status report to a receiving ASP. A delivered status report contains an identifier of the reported message.

A DELIVER.Response is issued by the receiving ASP to indicate whether it has taken responsibility of that message or could not accept responsibility for that message.

NOTIFY.Indication/Response (NTIN/NTRS)
The support of the NOTIFY service primitives has been dropped.

DISCONNECT.Request/Confirmation (DCRQ/DCCF)
A DISCONNECT.Request is issued by an ASP to free a previously acquired partner connection (conversation).

A DISCONNECT.Confirmation is the response of the MERVA Link message transfer layer to a DISCONNECT.Request. A DISCONNECT.Confirmation indicates whether a connection to the partner was successfully freed.

DISCONNECT.Indication/Response (DCIN/DCRS)
A DISCONNECT.Indication is issued by an MTP to allow a receiving ASP to disconnect from its application.

A DISCONNECT.Response is the response of the receiving ASP to a DISCONNECT.Indication.

MTP Boundary Function Service Primitives

The message transfer service processor communicates with a message transfer program using the following set of service primitives.

CONNECT.Request/Confirmation (CNRQ/CNCF)
A CONNECT.Request is issued by the message transfer service processor as response to the corresponding request from the ASP. Part of CONNECT.Request processing is the generation and transmission of a PROBE PDU to test for the availability of the partner ASP.

A CONNECT.Confirmation is the response of the sending MTP to a CONNECT.Request.

SendPDU.Request/Confirmation (SPDU/SPCF)
A SendPDU.Request passes a PDU to a sending MTP for transfer to the applicable partner.

A SendPDU.Confirmation is the response of the sending MTP to a SendPDU.Request.

ProcessPDU.Indication/Response (PPDU/PPRS)
A ProcessPDU.Indication is issued by a receiving MTP to pass a PDU to the message transfer service processor.

A ProcessPDU.Response is the response of the message transfer service processor to a ProcessPDU.Indication.

DISCONNECT.Request/Confirmation (DCRQ/DCCF)
A DISCONNECT.Request is issued by the message transfer service processor as response to the corresponding request from the ASP.

A DISCONNECT.Confirmation is the response of the sending MTP to a DISCONNECT.Request.

DISCONNECT.Indication/Response (DCIN/DCRS)
A DISCONNECT.Indication is issued by a receiving MTP to indicate that it will terminate the process.

A DISCONNECT.Response is the response of the message transfer service processor to a DISCONNECT.Indication.

Part 2. MERVA Link ESA Application Support

Chapter 5. Application Support Concepts and Resources

MERVA Link Application Support is implemented in MERVA ESA application programs. The sending Application Support Process is called by MERVA ESA via its transaction code in the MERVA ESA function table. Once in control, a MERVA Link ASP connects to MERVA ESA and requests services from MERVA ESA.

The interactions between MERVA Link and MERVA ESA follow the rules of MERVA ESA macro interface programming, which are described in the *MERVA for ESA Macro Reference*.

This chapter describes:

- The MERVA Link application control queue (ACQ).
- The Application Support (AS) status and the Message Transfer (MT) status of a MERVA Link sending ASP. The AS status of an ASP is identified by an alphabetic status identifier. The MT status of and ASP is identified by a numeric status code. Both parts of the sending ASP status determine whether MERVA Link should process outgoing messages, and if so, how.
- The MERVA Link message class concept. A message class is assigned to any message processed by MERVA Link in various states of the message transfer process to identify that state. This message class is important information for the correct routing of a message.
- The MERVA Link Application Support control fields, which are used by MERVA Link to control the message transfer process in MERVA Link programs, in MFS user exits, and in routing tables.

MERVA Link Application Control Queue (ACQ)

Each MERVA Link ASP has its own application control queue (ACQ) for its internal processing purposes. An ACQ cannot be shared among ASPs. Each ACQ:

- Contains an LC control message, if a sending ASP was once active. The LC control message is used to record the status of a sending ASP, and to control the message integrity of outgoing messages.
- Contains an LR control message, if any message was received by the receiving ASP. The LR control message is used to control the message integrity of incoming messages.
- Can contain one or more in-process (IP) messages when a sending ASP is processing messages. IP messages can be in the ACQ of an inactive ASP if an error was found in a sending process.

MERVA Link Sending ASP AS Status

The AS status of a MERVA Link sending ASP deals with the link of an ASP with its message source, it can be one of:

- OPEN-NOHOLD
- OPEN-HOLD
- CLOSED-NOHOLD
- CLOSED-HOLD

The MERVA System Control Facility command:

- **hold** *asp* puts an ASP into a HOLD status. An ASP in a HOLD status cannot and will not read any message from a queue of its send queue cluster.
- **astart** *asp* puts an ASP in a NOHOLD status. An ASP in a NOHOLD status can read messages from a queue of its send queue cluster.
- **aclose** *asp* puts an ASP in a CLOSED status. An ASP in a CLOSED status will route any message read from its send queue cluster to another queue (or other queues) as specified by the routing table associated with its application control queue. It will not (try to) transmit messages read from the send queue cluster to its partner. An ASP can be changed from OPEN to CLOSED only if it is in a HOLD status.
- **aopen** *asp* puts an ASP in an OPEN status. An ASP in an OPEN status can attempt transmitting messages read from a queue of its send queue cluster to its partner.

AS Status OPEN-NOHOLD

The AS status OPEN-NOHOLD is the initial and the normal AS status of an ASP. An ASP in this status can transmit messages to its partner and it is able to read messages from its send queue cluster. The MT status of the ASP, which is described later in this chapter, determines, however, whether actual message transmission is possible or not. The AS status OPEN-NOHOLD is a prerequisite for the successful transfer of messages in the send queue cluster to the partner ASP.

An **aopen** command for an ASP in this status is not accepted. It would have no effect. The effect of a START or a **kickoff** command for an ASP in this AS status depends on the current MT status of the ASP. Refer to the description of the **astart** and **kickoff** commands of the MERVA System Control Facility in the *MERVA for ESA Operations Guide*.

A **hold** command for an ASP in this status has the following effect: The ASP is put to HOLD status, the queues of the send queue cluster are set to HOLD, processing of the messages in the current transmission window is finished, and message transmission stops. Messages in the send queue cluster wait for transmission until the ASP is started again using the command **astart**.

An **aclose** command for an ASP in this status is not accepted. To close the ASP, set it to HOLD status before issuing the **aclose** command.

AS Status OPEN-HOLD

The AS status OPEN-HOLD is obtained from the initial AS status by issuing a **hold** *application* command. An ASP in this status collects messages in its send queue cluster. It does not transmit these messages.

An **aopen** command for an ASP in this status has no effect. The same applies for a **hold** command.

An **astart** command for an ASP in this status has the following effect: The ASP is put to NOHOLD status and the first queue of the send queue cluster is started. This results in a start of the applicable ASP that resumes message transmission.

A **kickoff** command for an ASP in this status has the following effect: The ASP is kept in HOLD status. It is started and resumes transmission of IP messages in the application control queue. Messages in the send queue cluster are not transmitted. If there are no IP messages in the application control queue, this **kickoff** command has no effect.

An **aclose** command for an ASP in this status puts it into CLOSED status.

AS Status CLOSED-NOHOLD

The AS status CLOSED-NOHOLD cannot be obtained directly from the initial status. It can be obtained from the status OPEN-NOHOLD by issuing the command sequence **hold**, **aclose**, and **astart**.

An ASP in this status routes messages in its send queue cluster (message class RS) immediately to a queue or to queues specified by the routing table associated with the application control queue. It does not transmit these messages to the partner ASP.

An **aclose** command for an ASP in this status is not accepted. A **kickoff** or an **astart** command for an ASP in this AS status starts the ASP for routing (not for transmission) of the messages in the send queue cluster.

A **hold** command for an ASP in this status has the following effect: The ASP is put to HOLD status, the queues of the send queue cluster are set to HOLD, and message routing stops. Messages in the send queue cluster wait for routing until the ASP is started again using the command **astart**.

An **aopen** command for an ASP in this status is not accepted. To open the ASP, set it to HOLD status before entering the **aopen** command.

AS Status CLOSED-HOLD

The AS status CLOSED-HOLD is obtained from the initial status by issuing a **hold** *application* and an **aclose** *application* command. An ASP in this status collects messages in its send queue cluster. It does not route or transmit these messages.

An **aclose** command for an ASP in this status has no effect. The same applies for a **hold** command.

An **astart** command for an ASP in this status has the following effect: The ASP is put to NOHOLD status and the first queue of the send queue cluster is started. This results in a start of the applicable ASP that resumes message routing to a queue or to queues specified by the routing table associated with the application control queue. Messages are not transmitted to the partner ASP.

A **kickoff** command for an ASP in this status starts the ASP but keeps it in HOLD status.

An **aopen** command for an ASP in this status has the following effect: The ASP is put to OPEN status, and messages in the send queue cluster are transmitted to the partner ASP. This transmission starts as soon as the ASP is set to NOHOLD status and the ASP is started by the **astart** command.

MERVA Link Sending ASP MT Status

The MT status of a MERVA Link sending ASP deals with actual message processing (transmission or routing). The following ASP status codes are used to describe the MT status:

00 The general meaning of this status code is *confirmed* or *delivered*. The transfer of all messages processed so far has been confirmed by the partner application. The message transfer is complete and the messages have been locally routed as required.

04 The general meaning of this status code is *accepted*. The request to transfer a message was accepted and no error was found in the local system when processing this request. However, an explicit confirmation from the remote system is outstanding. This is a normal status for all not-last messages in a transmission window.

08 The general meaning of this status code is *error detected in the local system*. This error possibly requires the intervention of the system administrator.

Examples for this status are situations where the link to the partner system is not in service or where a local Message Integrity Protocol violation has been detected.

09 The general meaning of this status code is *error detected in the remote system*. This error most probably requires an intervention by the system administrator in either the local or remote system, or in both systems.

A Message Integrity Protocol violation that is detected in the remote system, for example, is reported in the local system by this code. Another reason for this status code might be that MERVA ESA is not active in the remote system.

12 The general meaning of this status code is *severe error detected in the local system*. A MERVA Link customization error or an error in the local MERVA Link system is reported by this status code.

13 The general meaning of this status code is *severe error detected in the remote system*. A severe error in the remote MERVA Link system is reported by this status code.

| An ASP in the status 08, 09, 12, or 13 cannot transmit messages to its partner ASP.
| It can be manually started via the MERVA System Control Facility commands
| **astart**. It might be automatically started if the ASP monitor is active, or if
| **START=RETRY** has been specified in the EKAPT ASP entry.

MERVA Link Message Class Concept

MERVA Link uses a concept of message classes to provide its AS services. The class of a message is identified by the contents of the MERVA Link control field EKACLASS in the MERVA ESA TOF, which consists of 2 alphabetic characters. The field EKACLASS is located at nesting level 0 in the MERVA ESA TOF.

A message class is assigned by MERVA Link to any message that is processed by MERVA Link. The class of a message can change as it is routed within a MERVA ESA system. The class of MERVA Link internal control messages is, however, not changed.

MERVA Link requires an application control queue (ACQ) to be defined for any ASP. This queue can contain messages of three different classes (LC, LR, and IP). MERVA Link accesses messages in an ACQ via KEY1. Any MERVA Link ACQ must therefore be defined with the field EKACLASS as KEY1.

The following message classes are defined by and reserved for MERVA Link:

LC The class of the MERVA Link internal LC control message. A single LC control message is contained in the ACQ of any ASP that was once active. It is generated automatically and updated by a MERVA Link ASP.

The main purpose of the LC control message is to hold the message sequence number (MSN) of the application or acknowledgment message that was confirmed last. Confirmed in this context means that the sending MERVA Link system was informed of the successful delivery of the message to the intended receiving application.

The LC control message holds, in addition, other ASP-specific control information, the ASP status (AS and MT status), the MIP window size for ASPs associated with an APPC-type MTP, and a short note explaining the ASP's MT status.

LR The class of the MERVA Link internal LR control message that is contained in a MERVA Link ACQ. LR is also the class of any message received from a partner system. Finally, LR is the class of any message sent and correlated with an incoming acknowledgment message.

Messages of the class LR can therefore be contained in ACQs, ACK wait queues, completed message queues, and in received message queues.

IP The class of a message in the process of transfer to the intended receiving application. Message class IP is assigned to any message read from a send queue before it is written to the applicable ACQ and deleted from the send queue.

The number of IP messages in an ACQ is topped by the MIP window size, which is specified in the LC control message. IP messages must remain in that sequence in the ACQ in which they were written into this queue. The correct sequence is checked as specified by the message sequence number in the TOF field EKAAMSEQ of an IP control message.

CF The class of a confirmed message. When a sending MERVA Link system is informed about the successful delivery of one or more messages to the intended receiving application, it routes the delivered application messages as messages of the class CF to an ACK wait queue or to a complete message queue (if no acknowledgment is expected). Confirmed acknowledgment messages must be discarded or routed to a confirmed acknowledgment message queue.

CA The class of a confirmed and acknowledged message. An ASP can receive an acknowledgment for a message that has not yet been confirmed. The subject message is still in the ACQ as an IP message when the acknowledgment message is received. In that case, the control information in the acknowledgment message is added to the IP message in the ACQ, and the IP message remains in the ACQ until it is confirmed.

When the sending MERVA Link system is informed about the successful delivery of that message to the intended receiving application, it routes the delivered and acknowledged message with the class CA to an ACK wait queue or to a complete message queue (if the final acknowledgment has already been received).

RC The class of an IP message copied from the ACQ of an inoperable or closed ASP via the MERVA System Control Facility command **recover**. The routing table associated with the ACQ must identify a recovered and copied message by its class and take appropriate action.

Messages of the class RC always contain the Possible Duplicate Message (PDM) indicator.

RI The class of an IP message that could not be successfully delivered to the intended receiving application and was routed out of the transfer process to continue transferring other messages. This routing was requested via the MERVA Link Control command **iprecov** or performed, automatically, by the MERVA Link Automatic IP Message Recovery Function.

The routing table associated with the ACQ must identify a recovered IP message by its class and take appropriate action.

| **RM** The class of an IP message that was immediately removed from the outbound window upon request from an ASF. The message has not been sent to the partner ASP. The routing table associated with the ACQ must identify such a message by its class and take appropriate action.

RS The class of a message that is rerouted from the send queue of a closed ASP. The routing table that is associated with the ACQ must identify a rerouted message by its class and take appropriate action.

RR The class of a message that is rerouted from the send queue upon the request of the MERVA Link sample ASP user exit. The routing table that is associated with the ACQ must identify a rerouted message by that class and take appropriate action.

The message class RR is to be considered as a MERVA Link sample. It is not defined by and reserved for MERVA Link.

MERVA Link Application Support Control Fields

MERVA Link supports a number of control fields in MERVA ESA messages for both outgoing and incoming messages. Message transfer control information can be provided, for example, in MERVA Link control fields that are part of the message in the TOF. MERVA Link control fields are added to any message processed by MERVA Link. These fields contain message transfer parameters, for example, the originator and the recipient address, the submit timestamp, and the MERVA Link message identifier.

MERVA Link control fields are part of a MERVA ESA message on the MERVA ESA queue data set and part of the message in the MERVA ESA TOF. These control fields are part of a screen, printer, or network format of a message only if the corresponding message format definition in a MERVA ESA MCB asks for these fields. Messages formatted to SWIFT format do not contain any of the MERVA Link control fields.

The MERVA Link control fields are written to the MERVA ESA TOF at nesting level 0.

The main purpose of the MERVA Link control fields is to provide:

- A means to pass message transfer control information to MERVA Link
- Control information to be referenced by MERVA ESA routing tables
- Control information for message transfer control applications
- Control information for MERVA Link internal processing

The names of all MERVA Link control fields start with the MERVA Link component identifier EKA as the field name prefix. A specific field is identified by the five letters following the prefix, called the short field name. The short field name, however, can be customized via a modification of the EKAPT macro (rather than via customization parameters).

The following description of the MERVA Link control fields uses the short field names as they are provided by MERVA Link as defaults in the partner table header.

The names of the MERVA Link control fields (excluding the prefix EKA), the identifiers of the corresponding data elements (if applicable), and the names of these control data items are contained in the following table:

Name	DE ID	Control Data Item Name
AMCID	9200	ASL Message Identifier (Content ID)
MSGID	9201	MTL Message Identifier
AMSID	9202	IAM Message ID
AMSEQ	9203	MIP Message Sequence Number
MIPID	9204	MIP Message Identifier
IMSEQ	----	MIP Message Sequence Number (inbound)
OAFFN	A001	Originating Application Free Form Name
ONODE	A101	Originating MT Node Name
OAPPL	A102	Originating Application Name
MTPNM	----	Internal Message Transfer Process Name
RAFFN	A001	Receiving Application Free Form Name
RNODE	A101	Receiving MT Node Name
RAPPL	A102	Receiving Application Name
NETID	9602	MERVA ESA Line (Net) Format Identifier
AMBSL	9608	Buckslip
AMSUB	9609	Message Subject
SUBDT	----	SUBMIT Timestamp
SUBRC	----	SUBMIT Return Code
SUBDC	----	SUBMIT Diagnostic Code
DELDT	9301	DELIVER Timestamp
DELRC	9501	DELIVER Return Code
DELDC	9502	DELIVER Diagnostic Code
RECDT	9301	RECEIPT Timestamp
RECRC	9501	RECEIPT Return Code
RECDC	9502	RECEIPT Diagnostic Code
RDATA	9503	RECEIPT Report Data
TARQD	9610	Application Defined Request Data
TARSD	9611	Application Defined Response Data
TAACK	9612	Application Defined Acknowledgment Data
TAMAC	9613	Application Defined MAC Data
TAPAC	9614	Application Defined PAC Data
ACKRQ	B001	Request for ACK (Receipt Confirmation)
PRIOR	B002	Message Priority
PDUPM	C001	PDM Indicator
CLASS	----	MERVA Link Message Class
AWQSN	----	ACK Wait Queue QSN
WSIZE	9604	MIP Window Size
ACQNM	----	Name of the applicable ACQ

The three fields (CLASS, AWQSN, and WSIZE) are for MERVA Link internal purposes. They are used, for example, to control the integrity of the message transfer according to the rules of the Message Integrity Protocol. The field ACQNM is provided to be used by a routing table to route a message to the applicable MERVA Link ACQ.

Chapter 6. Application Support Functions

This chapter discusses the logic how MERVA Link retrieves a message from the applicable send queue cluster, how it determines whether the retrieved message is an outgoing message or a status report (acknowledgment message), and how either of these messages is processed.

Other items discussed in this chapter are the correlation of acknowledgment messages with the reported message and the handling of delivery errors.

Sending Messages

A message is passed to MERVA Link by routing it to a MERVA ESA queue that is a member of a MERVA Link send queue cluster. The MERVA Link ASP gets the messages from the send queue cluster as specified by the priority of the queues in this cluster (specified sequence of the queues in the cluster definition in the partner table).

A message in a MERVA Link send queue can be an application message or a status report (acknowledgment message). An application message is sent to the intended partner application via an IM-ASPDU, whereas an acknowledgment message is sent via an SR-ASPDU. This is why the MERVA Link ASP needs to know whether the retrieved message is an application message or an acknowledgment message.

The MERVA Link ASL Interface defines that an outgoing message that contains the control field *EKARECRC* with a valid receipt return code is an acknowledgment message. All other messages are treated as application messages.

Acknowledgment Control Information

The information in an SR-ASPDU that is related to the ACK is contained in the receipt-report data element (ID=1500). The receipt-report data element contains up to four types of lower-level data elements, the:

- Receipt Timestamp
- Receipt return code
- Receipt diagnostic code (optional)
- Receipt report data (optional)

The receipt-report-data data element can be contained in the receipt-report data element more than once.

Inserting Acknowledgment Information in Outgoing Messages

A new application or a modified existing application can add receipt control information to an acknowledgment message before it is routed to a MERVA Link send queue. However, it might not be possible to modify existing applications for providing this control data. A MERVA ESA MFS user exit enables MERVA Link to add MERVA Link receipt control information to an acknowledgment message.

The number of a MERVA ESA MFS user exit is associated with an ASP in the partner table. This user exit is called when an outgoing message has been retrieved from the send queue and has been formatted to the MERVA ESA TOF format.

This user exit checks whether a message is an application message or an acknowledgment message. In case of an acknowledgment it adds MERVA Link receipt control information to the message in the MERVA ESA TOF. A sample MFS user exit is provided by MERVA Link.

The MERVA Link Application Support Processor finds and processes the receipt control information.

Receiving Application Messages

A message is passed to a MERVA ESA application by routing it to one or several MERVA ESA queues as specified by a MERVA ESA routing table. One of the twelve possible destination queues is reserved for MERVA Link MIP processing. A received message must always be routed to the applicable MERVA Link ACQ.

The MERVA ESA application can return an acknowledgment to the originator of that message, based on that incoming message. This acknowledgment message is sent to the partner application in the same way as an application message.

An incoming application message can be handled by a MERVA ESA MFS user exit before it is routed to the destination queues. The applicability of an MFS user exit is specified in the partner table ASP entry.

Receiving Acknowledgment Messages

A MERVA Link Acknowledgment Message (ACK) is a message containing a valid receipt return code in the field EKARECRC. It is constructed from information in an SR-ASPDU. An acknowledgment message is either merged with the applicable reported message and routed as appropriate, or it is immediately routed as appropriate. The format of the message in the latter case is that of the MERVA Link control message MCTL. This message type is specified by the EKAMCTL Message Control Block.

The logic how an incoming acknowledgment message is merged with the applicable reported message is described in the following sections.

Merging an ACK with the Original Message

An acknowledgment message is sent by an application, which has received an application message from its partner, back to this partner in an SR-ASPDU. The application message originating ASP receives the acknowledgment message and correlates this ACK with the reported application message. Correlation means to find the applicable reported message in the MERVA Link ACK wait queue or in the MERVA Link control queue, to add the receipt report data to this message, and to route the acknowledged message as appropriate.

Correlation Data

The MERVA Link IAM message ID serves for correlation of an acknowledgment message with the applicable reported message. The MERVA Link IAM message ID can be contained in a MERVA Link control field provided by an application. If an application provides an IAM message ID, it is used by MERVA Link for the correlation of an ACK with its reported application message. Otherwise, MERVA Link generates an IAM message ID for that purpose.

Application Message in the ACK Wait Queue

An application message has been routed to the applicable ACK wait queue when the transfer of the message to the recipient application was confirmed. The ACK wait queue has been defined in the MERVA ESA Function Table with the IAM message ID control field as key. A MERVA Link Application Support Program can therefore retrieve a specific message from the ACK wait queue based on the IAM message ID.

ACK Correlation Process

The ACK correlation process is initiated when a receiving ASP gets a receipt report (SR-ASPDU). After checking message integrity, it extracts all applicable report information. One item of this information is the correlation data (IAM message ID).

The reported application message is retrieved from the ACK wait queue using the correlation data. When this message has been found in the ACK wait queue, receipt-report information is added to that message, and the modified message is routed as appropriate.

ACK Correlation Failure

The reported message might not have been found in the ACK wait queue when a status report was processed that should have been correlated with the reported message. This can occur for the following reasons:

- More than one final ACK sent by the recipient of the reported message
- A MERVA Link customization error
- An error in a routing table involved in MERVA Link processing
- A delivery report processing error at the sending or receiving side
- A race condition where the status report is processed at the originally sending side before the delivery confirmation of the reported message has been completely processed

To handle the race condition between the delivery confirmation and the status report, MERVA Link waits for the sending ASP to complete the current window, then tries again to find the reported message in the ACK wait queue. If the reported message is found during this attempt, it is processed as described above.

If the reported message is, again, not found in the ACK wait queue, MERVA Link tries to find the reported message as IP message in the application control queue (ACQ). If the reported message is found in the ACQ:

- The acknowledgment control information is saved in the IP message, the receipt return code 00, 04, or 08 is masked to 01, 05, or 09, respectively, and the updated IP message is replaced in the ACQ.
- A general control message (type MCTL) with message class LR is generated from information in the received acknowledgment message, and is routed to the ACQ as the new LR control message replacing the old LR control message.

If the reported message is not found, a general control message (type MCTL) with message class LR is generated from information in the received acknowledgment message and routed as appropriate for an inbound message.

MIP Considerations during ACK Processing

The MERVA Link MIP requires that a MERVA Link ASP incoming message is routed as the new last received (LR) control message to the application control queue (ACQ), and to the applicable destination queues (maximum eleven queues). At the same time, this means that, with the same MERVA ESA queue management request, the old LR control message must be deleted from the ACQ.

ACK processing, however, requires that the acknowledged message, not the incoming message (which is also the acknowledgment message), be routed. MERVA ESA does not support a service request that handles all four activities required by concurrent MIP and ACK processing at once.

This problem is solved by an extension of MIP processing, and a second step when processing an ACK as follows:

When the receipt report information has been added to the reported message, the following MIP-related information is added to this message in MERVA Link control fields:

- Message Class LR
- MIP MSN of the ACK
- MERVA ESA queue sequence number of the acknowledged message in the ACK wait queue

A single MERVA ESA Queue Management request routes the acknowledged message to the ACQ and to the applicable destination queues, and deletes the old LR control message from the ACQ. Another MERVA ESA Queue Management request deletes the acknowledged message from the ACK wait queue if the reported message was found in that queue.

To recover from a possible system failure between the latter two steps, MIP processing for a received message is extended as follows.

When the LR control message has been retrieved from the ACQ, the message identified by the queue sequence number contained in the MERVA Link control field EKAAWQSN (ACK wait-queue sequence number) is deleted from the ACK wait queue. In most cases, that is, when the subject failure did not happen, no message is deleted from the ACK wait queue. When the last received message was no ACK, the control field EKAAWQSN will not contain a valid MERVA ESA queue sequence number and deletion of a message will not be requested.

Automatically Starting an Inoperable ASP

When a MERVA system comes up in the morning and an ASP starts sending messages to its partner, the ASP becomes inoperable if the partner system is not yet active. Two functions are provided to recover the sending ASP from that inoperable status.

When the partner system comes up and sends a message (PROBE or a real message), the receiving ASP (associated with the inoperable sending ASP) can kick off the sending ASP. When another message is routed to a send queue of an inoperable ASP, the ASP can retry to start the sending process despite its inoperable status.

Sending an ASP Kickoff from a Receiving ASP

An inoperable (sending) ASP with AS status OPEN/NOHOLD and MT status 08 or higher is automatically started at the end of a receiving ASP task (immediately before the receiving ASP disconnects from MERVA ESA). Hours and minutes of the current time must be different from the status time in the LC control message to avoid multiple starts within a short time frame.

A remote inoperable ASP can be manually kicked off if the ASPs are connected through APPC-type MTPs. A **kickoff** command for such an ASP causes a PROBE to be sent to the remote partner ASP, which might be inoperable. If the partner ASP is indeed inoperable, it is kicked off by the receiving ASP in the remote system when the PROBE has been successfully processed.

For more information about the **kickoff** command, refer to the MERVA System Control Facility in the *MERVA for ESA Operations Guide*.

Retry to Start an Inoperable Sending ASP

An ASP customization option in the partner table (**START=RETRY**) asks an inoperable ASP to retry the transmission of the messages in its send queue cluster when another message has been routed to one of its send queues. This request for automatic retry is honored only if hours and minutes of the current time are different from the status time in the LC control message to avoid multiple starts within a short time frame.

ASP Monitor

An ASP monitor is supported in the MERVA ESA CICS environment. It can automatically start an inoperable ASP without an external event.

The ASP monitor is a MERVA Link program that periodically scans the partner table for inoperable ASPs, and provides the same functionality as a **kickoff** command for all eligible ASPs. For more information about the MERVA Link CICS ASP monitor, refer to “Chapter 7. The MERVA Link CICS ASP Monitor” on page 77.

Handling Message Delivery Errors

When a message is sent to a receiving ASP in net format it is formatted to TOF format. This formatting process can fail or return a warning.

An MFS user exit is called before an incoming message is routed to the destination queue(s). It indicates whether the message is correct or not correct.

Message delivery error handling applies if the formatting process or the MFS user exit detect any problem (warning or error).

Two functions are provided to handle a message delivery error. The first function allows a receiving ASP to accept a message despite a warning. The second function allows a sending ASP to recover automatically from an IP-message that cannot be successfully delivered to its partner receiving ASP. This function is also provided as an operator command through the MERVA System Control Facility (MSC).

Accept an Inbound Message Despite a Warning

A parameter of the ASP definition in the MERVA Link partner table tells whether an inbound message that could not be formatted from net format to TOF format (or queue format to TOF format) without a warning must be accepted by MERVA Link and be passed to the receiving application. The same applies for a warning returned by the user exit.

The warning codes of a formatting error (MFS return code 04 and the MFS reason code) are recorded by MERVA Link in the delivery report fields EKADELRC and EKADELDC. The delivery return code (field EKADELRC) contains 04, and the delivery diagnostic code (field EKADELDC) contains the 4-byte character representation of the hexadecimal MFS reason code (padded with two blanks to fill the 6-byte diagnostic code field).

The user exit and the routing table can react upon these delivery error codes. As its reaction, the user exit can, for example, modify the delivery report fields EKADELRC and EKADELDC. And a routing table can route a message, which was accepted despite a warning, to a manually processed error queue.

A message that is accepted by a receiving ASP is reported to the sending ASP as having been “delivered without error”.

Recovering from Delivery Errors

When a receiving ASP detects a delivery error, it notifies the sending ASP, and in the notification identifies the corresponding IP-message. A delivery error reported by a receiving ASP causes the sending ASP to be set inoperable. This event is reported to the console operator via message EKA701E. Recovering from this situation entails doing the following:

- Changing the class of the message from IP to RI (an abbreviation of ″**r**ecovered **I**P message″), to indicate that it could not be delivered, and routing the message as specified by the routing table associated with the ACQ
- Setting the ASP status to **operable** with status code 00 and diagnostic code AIPRCV (an abbreviation of ″**a**utomatic **IP** message **r**e**c**o**v**ery successful″)
- Resequencing the message numbers (MSNs) of any subsequent messages in the ACQ so that there are no gaps between MSNs
- Resuming transmission with the next message

Due to the resequencing of MSNs, several non-delivery notifications might refer to the same MSN, namely when several attempts to transmit messages fail one right after the other. Do not be confused by this. When the MIP window size is not 1, you can see on the Display Specific ASP/MTP (AC02) screen that the number of IP messages in the ACQ is changing.

A parameter of the ASP definition in the MERVA Link partner table determines whether the sending ASP tries to recover from non-delivery situations automatically, or waits for someone to address this situation manually.

Automatic Recovery

If automatic recovery was specified for a sending ASP, the steps listed above are carried out automatically. In addition, during automatic recovery, the LC MSNs in the LC control messages of messages that were successfully delivered are updated accordingly. This prevents the retransmission of any already delivered messages when message transmission is resumed. Because an ASP transfers all messages sequentially, and because the MSN of a message that could not be delivered is reused for the next message, a non-delivery notification for a specific IP-message implies that all preceding messages (that is, all messages with lower MSNs) were delivered successfully.

If the non-delivery notification does not adequately identify the affected IP-message, or if the reported message cannot be found in the ACQ, or if there is an error during the automatic recovery process, then the process terminates, and the original SUBMIT/DELIVERY error is reported as if automatic recovery were not requested.

After automatic ASP recovery is performed and is successful, the ASP is operable again, and the console operator is informed via message EKA703I that no action is required to address the message EKA701E that was issued when the ASP was set inoperable.

Manual Recovery

If automatic recovery was not specified for a sending ASP, the recovery steps listed previously must be carried out manually. To do this, on the Display Specific ASP/MTP (AC02) screen enter the MERVA System Control Facility (MSC) command **iprecov** ***msn***, where ***msn*** is the message number of the undelivered IP message.

The **iprecov** command is accepted only if it is specified in the AC02 screen. This ensures that the applicable ASP and the ACQ that contains the subject IP message are clearly identified. This screen also shows the LC MSG note that contains the MSN of the IP-message that was not delivered. This MSN must be specified as parameter of the **iprecov** command.

The **iprecov** command is accepted only for an inoperable ASP. It includes the functionality of a **kickoff** command, which is described briefly in "Sending an ASP Kickoff from a Receiving ASP" on page 72, and in more detail in the MERVA System Control Facility in the *MERVA for ESA Operations Guide*.

Unlike during automatic ASP recovery, the **iprecov** command does not update the LC MSN , and so some already delivered messages might be retransmitted. This retransmission, however, does not mean that the message is accepted by the receiving application a second time: the MIP at the receiving side recognizes if a message has already been sent, and discards such messages.

A consequence of manual recovery is that the delivery return and diagnostic codes are not available. To give an indication of the manual IP message recovery in the recovered message, the delivery return and diagnostic codes are set to 08 and IP RCV, respectively. The delivery return code 08 indicates that the diagnostic code IP RCV has been provided by the local system rather than by the partner system.

Recovering from a Recovery Process Interrupt

Because the MSNs of IP messages in an ACQ must be an ascending sequence without gaps, when an IP message is routed out of the ACQ during automatic or manual recovery, the subsequent IP messages are assigned new message sequence numbers (MSNs). If the process of assigning new MSNs is interrupted (for example, if MERVA is terminated during this process), a gap remains in the sequence of MSNs. When the MERVA Link sending task is started again, it reports the gap as a local MIP violation. To recover from such a violation:

1. Make sure the ASP is inoperable.
2. On the Display Specific ASP/MTP (AC02) panel, issue the command **iprecov** *.

|
Immediate Recovery

| An application support filter can check an outbound message and, if the message
| is not to be sent to the partner ASP, arrange for it to be instead rerouted as

specified in the routing table associated with the ACQ. The ASP continues sending messages without an error indication. The class of this message when it is passed to the routing table is set to RM.

Chapter 7. The MERVA Link CICS ASP Monitor

An ASP monitor is provided in the MERVA Link CICS environment. Once started, this program checks all ASPs defined in the MERVA Link partner table (PT) periodically for inoperable ASPs. It issues a **kickoff** command for every inoperable ASP that meets a number of additional criteria. The ASP monitor is not supported in the MERVA Link IMS environment.

Operating the ASP Monitor

The ASP monitor can be started automatically by the first executing entity of any sending or receiving ASP. As an alternative, it can be started and stopped manually by means of the MSC **set** and **reset** commands, or by the explicit call of the ASP monitor transaction EKAM.

The ASP monitor restart time-interval can be set from one minute to 59 minutes when the ASP monitor is started, and can be manually reset to any value from 01 to 59 at any time. All manual modifications of the restart time interval are applicable until a new PT is loaded from the module library. At the begin of a CICS job, the ASP monitor parameters defined in the PT are applicable.

Automatic ASP Monitor Start

The permanent characteristics of the ASP monitor can be defined in the MERVA Link partner table. The transaction identifier of the ASP monitor task and the ASP monitor task restart time interval can be specified in the **MONITOR** parameter of the **EKAPT TYPE=INITIAL** macro instruction.

When a valid transaction code (for example, EKAM) and a restart time interval different from 00 are specified in the **MONITOR** parameter of the **EKAPT TYPE=INITIAL** macro instruction, the ASP monitor is automatically started by the first sending or receiving ASP task in a CICS job. All further modifications of the ASP monitor task (for example, stop, restart, reset time interval) must be performed manually, if required.

ASP Monitor Handling within MSC

The MERVA System Control Facility provides a set of commands to modify the ASP monitor parameters. It provides also a means to monitor the execution of the ASP monitor.

MSC Commands for the ASP Monitor

When a valid transaction code and a restart time interval of 00 are specified in the PT, the ASP monitor is not automatically started. It can be, however, started and stopped by means of the MSC **set** and **reset** commands.

SET AM

The MSC command **set am** starts the ASP monitor if the ASP monitor transaction identifier and a restart time interval different from 00 are contained in the PT. The ASP monitor must, however, not be stopped before via EKAM STOP (see “Direct ASP Monitor Transaction Call” on page 78).

SET AM00
The MSC command **set am00** sets the restart time interval to zero. The effect is that the ASP monitor task is not restarted after its next instance.

SET AM01 .. AM59
The MSC commands **set am01** to **set am59** set the restart time interval to the number of minutes specified in the SET command parameter (01 to 59). The effect is that the ASP monitor task is started when the specified time interval has elapsed. If another instance of the ASP monitor task is scheduled within that time interval, it terminates without processing. This means, the new restart time interval becomes immediately active.

A SET AMnn command with nn outside the range 00 to 59 is processed like the SET AM command.

RESET AM
The MSC command **reset am** sets the restart time interval to zero. It has the same effect as the SET AM00 command.

Monitoring the ASP Monitor

A number of ASP monitor parameters and ASP monitor activity information are contained in the PT header. This information can be displayed via the MSC command **dpth** (for **d**isplay **p**artner **t**able **h**eader).

For more information about this display refer to the description of the MSC screen AC04.

Direct ASP Monitor Transaction Call

When no ASP monitor support is generated in the PT (this means, the **MONITOR** parameter of the **EKAPT TYPE=INITIAL** macro instruction is not specified), the ASP monitor is not automatically started. It can be, however, started and stopped by means of a direct call of the ASP monitor transaction (for example, EKAM).

EKAM START
The direct call of the ASP monitor transaction with parameter START starts the ASP monitor immediately and sets the restart time interval to five minutes. The ASP monitor transaction identifier in the PT is set to the transaction identifier entered as part of this direct transaction call (EKAM, in our example). The ASP monitor parameters in the PT are ignored by this task start request.

This ASP monitor task start request is also applicable when no ASP monitor parameters have been specified when the PT was generated, and when the ASP monitor was stopped via EKAM STOP.

EKAM STOP
The direct call of the ASP monitor transaction with parameter STOP stops the ASP monitor immediately. Any pending ASP monitor restart request results in an empty ASP monitor task.

The only means to restart the ASP monitor after EKAM STOP is the EKAM START direct transaction call (or a new PT in this CICS or in a new CICS job).

EKAM
The direct ASP monitor transaction call without a parameter starts the ASP monitor if the ASP monitor transaction identifier and a restart time interval different from 00 are contained in the PT. It corresponds to the MSC command **set am**.

EKAM 00
The direct ASP monitor transaction call with parameter 00 sets the restart time interval to zero. The effect is that the ASP monitor task is not restarted after its next instance. This corresponds to the MSC command **set am00**.

EKAM 01 .. 59
The direct ASP monitor transaction call with parameters 01 to 59 sets the restart time interval to the specified number of minutes (01 to 59). This corresponds to the MSC commands **set am01** .. **set am59**.

ASP Monitor Functions

| The ASP monitor follows a number of rules to evaluate whether an ASP must be
| started (kicked off).

ASP Kickoff Criteria

A number of criteria must be met before the ASP monitor issues a KICKOFF command for an ASP. As a matter of fact, it is actually a MERVA SF (start function) command for the first send queue of an ASP.

Automatic ASP Start Requested

The ASP monitor checks whether operator start is requested for this ASP (START=OPERATOR specified in the PT ASP entry). It disregards any ASP that is reserved for operator start.

Sending ASP Not Active

The ASP monitor checks whether an instance of the sending ASP task is currently active. It disregards an ASP if it finds any evidence that it is active.

ASP Not Closed or On Hold

The ASP monitor checks whether the ASP is open for message transmission and in no-hold status. It disregards an ASP if its status identifier in the LC control message is not ON (open, no-hold). An ASP without an LC control message in its ACQ is also disregarded. No error is reported in this case.

ASP Permanently Inoperable

The ASP monitor checks whether the ASP is inoperable. It disregards an ASP if its status code in the LC control message is smaller than 08.

ASP not Recently Active

The ASP monitor checks whether the ASP was active within the current minute. It disregards an ASP if its status timestamp in the LC control message (yymmddhhmm) is the same as the current timestamp. The seconds of the timestamp are ignored.

Restart Time Interval Considerations

A number of considerations apply when selecting the ASP monitor restart time interval. The minimum restart time interval of one minute means that the ASP monitor task is started once every minute. The additional load on the MERVA system by the ASP monitor itself can be ignored. It is very low, also if you have a large number of ASPs defined in the partner table.

The additional load on the local MERVA system by the sending ASP tasks started by the ASP monitor, and the possible additional load on the data communication network and on the partner MERVA systems must, however, be carefully considered. This additional load, and the fact that each ASP that remains inoperable writes the EKA701E operator message to the MVS (TM) console and the MERVA journal, suggest that the ASP monitor restart time interval should not be too small.

The maximum restart time interval of 59 minutes means that the ASP monitor task is started once about every hour. The additional load on the local MERVA system by the sending ASP tasks started by the ASP monitor, and the possible additional load on the network and the partner MERVA systems is at its minimum. The fact that messages ready for transmission possibly wait longer for transmission than necessary, suggests that the ASP monitor restart time interval should not be too big.

The optimum ASP monitor restart time interval depends on the characteristics of your environment. A restart time interval of five to ten minutes might be acceptable in the majority of the production environments. In a test environment, however, one minute is possibly the best choice.

Chapter 8. Support of MERVA ESA Facilities in MERVA Link

This chapter discusses the support of various MERVA ESA facilities by MERVA Link. These facilities are the MFS user exit, the routing table, the message trace, and the journal.

MERVA ESA MFS User Exit Support

A MERVA ESA MFS user exit can be associated with a MERVA Link ASP. This user exit is called by MERVA Link at six (logical) places for different purposes. These places and purposes are described in the following.

The rules how to write a MERVA ESA MFS user exit for use with MERVA Link is described in the *MERVA for ESA Customization Guide*

The rules how to write a MERVA ESA MFS user exit for use with MERVA Link is described in the *MERVA for ESA Customization Guide*. This description is based on the MERVA Link Sample MFS user exit EKAMU010.

A MERVA ESA MFS user exit for use with MERVA Link can access, modify, and write new MERVA Link control fields in the MERVA ESA TOF. These control fields are at nesting level 0. The user exit can handle TOF fields associated with other applications in the same way.

Ready-to-Send Messages

The MFS user exit is called by the sending ASP with function code S, for ready-to-send messages, before the message is moved from the send queue to the MERVA Link application control queue (ACQ).

The user exit tells MERVA Link to transmit the message, or to route the message immediately as specified by the routing table associated with the ACQ.

The MFS user exit must return a new message class in the first 2 bytes of the MFS user exit output data buffer (see parameter OUTBUF of the DSLMFS TYPE=USER macro) to request immediate routing out of MERVA Link. In response to this request, MERVA Link saves the specified message class in the MERVA Link control field EKACLASS in the message, and routes the message as specified by the routing table associated with the ACQ. This routing table can refer to the specified message class.

If the MFS user exit returns an empty MFS user exit output data buffer, MERVA Link will start the message transmission process.

The MFS user exit support at this place provides a means to close an ASP temporarily for a single message. For more information about a closed ASP refer to "MERVA Link Sending ASP AS Status" on page 61.

Outgoing Messages

The MFS user exit is called by the sending ASP with function code O, for outgoing messages, before the P2 PDU is constructed, which conveys the message to the partner. The PDU that will be constructed (an IM-ASPDU or an SR-ASPDU) depends on the nature of that message (an application message or an acknowledgment message).

It is the task of the user exit to tell MERVA Link whether the message is an application message or an acknowledgment message via data in the MERVA Link control field EKARECRC (receipt return code). A message containing 00, 04, or 08 in the field EKARECRC is processed as an acknowledgment message by MERVA Link. All other messages are processed as application messages.

When the MFS user exit is called with function code O, the message can contain 01, 05, or 09 in the field EKARECRC. This means that the message has been acknowledged before it was confirmed, and it must be retransmitted as application message to resynchronize the MERVA Link MIP. In this case, the user exit must not modify those fields in the message that are related to the acknowledgment. These fields are EKARECDT, EKARECRC, EKARECDC, and EKARDATA.

Depending on the originating MERVA ESA application, the correct acknowledgment control information can already be contained in an acknowledgment message passed to MERVA Link. The user exit might not have to do anything in this case (or even not be applicable).

Confirmed Messages

The MFS user exit is called by the sending ASP (APPC protocol) or by the receiving ASP (ISC protocol) with function code C, for confirmed (delivered) message, before the message is routed to the ACK wait queue or to a queue containing delivered messages.

When the transfer of an outgoing message has been confirmed or a delivery report has been received for that message, the user exit can identify this message as delivered to the recipient application in application specific terms. Both, application messages and acknowledgment messages can be controlled and modified by a user exit at this place.

The class of a message at this place is CF or CA. A message with class CA is a confirmed application message that contains already acknowledgment control information, this means, a valid receipt return code in EKARECRC. A valid receipt return code (00, 04, or 08) in a message of the class CF indicates an acknowledgment message. If it does not contain a valid receipt return code, the confirmed message with class CF is an application message.

Incoming Application Messages

The MFS user exit is called by the receiving ASP with function code I, for incoming application messages, before the message is routed to the destination queues.

It is the task of the user exit to check the message and to provide information that is used by the MERVA ESA routing table to route that message to the appropriate destination queues. If sufficient information is already contained in the message, the user exit might not have to do anything here.

Incoming Acknowledgment Messages

The MFS user exit is called by the receiving ASP with function code R, for incoming report (acknowledgment message), before an acknowledgment message or an acknowledged message (original message merged with acknowledgment message) is routed to the destination queues.

It is the task of the user exit to check the message and to provide acknowledgment information as appropriate for the destination application. If appropriate information is already contained in the message, the user exit might not have to do anything at this point.

When the MFS user exit is called with function code R, the message can contain 01, 05, or 09 in the field EKARECRC and message class IP in the field EKACLASS. This indicates a message that has been acknowledged before it was confirmed, and is still “in process”. The user exit must know that this IP message will be rerouted to the ACQ. It must not change the masked receipt return code in the field EKARECRC.

The masked receipt return code is unmasked to 00, 04, or 08 when the transfer of the message is confirmed. The message is then routed with class CA to the appropriate next queue (ACK wait queue or completed message queue depending on the receipt return code).

Recovered Messages

The MFS user exit is called by the ASP or by the MERVA System Control Facility with function code V when a message is recovered or routed out of MERVA Link because of a temporarily or permanently closed ASP. It is called before the message is journaled, and routed as specified by the routing table associated with the ACQ.

The class of a message at this logical place is one of the following:

RC Is the class of a message that has been recovered from an inoperable ASP via the **recover** command.

RI Is the class of a message that has been manually recovered from a delivery error via the **iprecov** command, or automatically by the ASP.

| **RM** Is the class of a message that has been immediately recovered from an
| outbound window upon request of an ASF.

RS Is the class of a message that has been routed out of MERVA Link because of a permanently closed ASP.

any Other message class can have been set by the user exit when it was called to check a message that is “ready to send”. By setting the message class to any value other than blanks, the user exit closes the ASP temporarily for this message.

A user exit called with function V can, for example, add a SWIFT PDE trailer to a message that carries a MERVA Link PDM indicator. The MERVA Link PDM indicator is added to a message when it is recovered from an inoperable ASP using the RECOVER command, and, for example, subsequently routed to the SWIFT ready queue.

Note: The message type is not passed to the user exit as an input parameter when it is called with function V. The message type can be contained in a TOF field.

MERVA ESA Routing Table Support

MERVA Link requests routing of a message as specified by a routing table associated with a MERVA ESA queue at six places. These places and the purpose of the destination queue(s) is described in the following.

A detailed description how to code a MERVA ESA routing table for the use with MERVA Link is described in the *MERVA for ESA Customization Guide* based on the MERVA Link sample routing tables EKARTS and EKARTS1I.

Route Ready-to-Send Messages to Another Send Queue

MERVA Link provides four facilities to route a message to a send queue of another transmission medium. This is useful if a message cannot be transmitted in time by MERVA Link or if it should not be transmitted by MERVA Link because of any other reason. These four facilities are:

- Processing of the messages in the send queue of a permanently closed ASP
- Message rerouting requested by the MFS user exit when it was called before MERVA Link message transmission processing started for a message in a send queue (temporarily closed ASP)
- In-process message recovery requested by the MERVA Link system administrator via the **recover** command for a permanently closed ASP
- Undeliverable message recovery requested by the MERVA Link system administrator via the **iprecov** command, or automatically performed by the MERVA Link sending ASP

The message class is:

RS	For messages rerouted from a send queue of a permanently closed ASP
RC	For in-process messages rerouted via the **recover** command from the ACQ of a permanently closed ASP
RI	For undeliverable messages recovered via the **iprecov** command from the ACQ of an inoperable ASP

A user-exit routine can determine whether a message is to be rerouted and, if so, set the message class so that the routing module will reroute it. Do not use the message classes LC, LR, IP, CF, CA, RC, RI, RM, or RS in a user exit routine for a rerouted message, as these are reserved for use by MERVA.

A rerouted message can be either an application message or an acknowledgment message. An acknowledgment message is not necessarily indicated by a valid receipt return code in the MERVA Link control field EKARECRC.

Route Confirmed Outgoing Messages

When the transfer of a message has been confirmed by the receiving ASP, it must be routed from the ACQ to an ACK wait queue (if an acknowledgment is expected for this message) or to a Transfer Process Finished queue (if no acknowledgment is expected).

The MERVA Link class of a confirmed message is CF for both an application message and an acknowledgment message. An acknowledgment message, however, must not be acknowledged by the receiver. Therefore, it must not be routed to an ACK wait queue. It can be routed, for example, to a MERVA ESA dummy queue in order to discard it.

The MERVA Link class of a confirmed message that already contains acknowledgment control information is CA. Dependent on the receipt return code, this message must be routed to the ACK wait queue (if more acknowledgment is expected) or to a Transfer Process Finished queue (if no more acknowledgment is expected).

Route Incoming Acknowledgment or Acknowledged Messages

When a receipt report has been correlated with the reported message and their information has been merged to an acknowledged message, or when an acknowledgment message has been generated from information in the receipt report, that message must be routed to the ACK wait queue (if another acknowledgment is expected for this message) or to a Transfer Process Finished queue (if the final acknowledgment has been obtained for that message).

The MERVA Link class of an acknowledgment message or an acknowledged message is LR. It must be routed to the applicable ACQ in addition to the destination queue(s). The ACQ name contained in the field EKAACQNM can be used by the routing table to route the received message to the ACQ as required.

Route Incoming Application Messages

An incoming application message can be routed to one or more incoming message queues. The maximum number of target queues supported by MERVA ESA in a message routing request is twelve. As MERVA Link asks for the ACQ to be one of these twelve target queues, the maximum number of target queues available for the receiving application is eleven.

The MERVA Link class of an incoming message is LR. It must be routed to the applicable ACQ in addition to the incoming message queue(s). The ACQ name contained in the field EKAACQNM of any incoming message can be used by the routing table to route the received message to the ACQ as required.

MERVA ESA Message Trace Support

MERVA Link adds entries to the MERVA ESA message trace to document MERVA Link processing in the message. There are eight places in MERVA Link programs where a message trace is written. All places where a message trace is written and the message trace data applicable at this place are described in the following.

Move Message from the Send Queue to the ACQ

MERVA Link starts message processing at the sending side by moving the highest priority message from the send queue cluster to the applicable ACQ.

This process is traced with the sending application support program name EKAAS10 as the user name, and the name of the applicable send queue as function name.

Route Confirmed Message (Synchronous Confirmation)

When the transfer of a message has been confirmed by the receiving system, the message is routed as specified by the outgoing message routing control information in the applicable PT ASP entry at the sending side.

This process is traced with the sending application support program name EKAAS10 as the user name, and the routing identifier of the outgoing message routing control information as function name.

Route Incoming Application Message

When an application message has been received, it is routed as specified by the incoming message routing control information in the applicable PT ASP entry at the receiving side.

This process is traced with the receiving application support program name EKAAR10 as the user name, and the routing identifier of the incoming message routing control information as function name.

Route Acknowledgment or Acknowledged Message

When a receipt report has been received, it is correlated with the original message or processed as a unique acknowledgment message. The acknowledged message or the acknowledgment message is routed as specified by the incoming report routing control information in the applicable PT ASP entry at the (original) sending side.

This process is traced with the receiving application support program name EKAAR10 as the user name, and the routing identifier of the incoming report routing control information as function name.

Route Ready-to-Send Message to Another Send Queue

When a message is routed from a MERVA Link send queue to the send queue of another message transmission medium because of a permanently or temporarily closed ASP, a message trace entry is written to the message.

This message trace entry contains the sending ASP program name EKAAS10 as the user name, and the name of the ACQ as the function name.

Copy In-Process Message to Another Send Queue

When an IP message is copied from a MERVA Link ACQ to the send queue of another message transmission medium using the **recover** command for an inoperable or permanently waiting closed ASP, a message trace entry is written to the message.

This message trace entry contains the name of the MERVA System Control Facility program name EKAEMSC as the user name and the name of the ACQ as the function name. The ASP MT status code applicable when the **recover** command is entered is shown as return code in the message trace entry.

MERVA ESA Journal Support

A number of events are journaled by MERVA Link in the MERVA ESA journal. These events can be divided into three classes. Any class contains journal entries of one or more journal entry types.

Classes of MERVA Link Entries in the MERVA ESA Journal

The events journaled by MERVA Link in the MERVA ESA journal can be divided into three classes:

- Events at the ASP/MTSP boundary
- Recovered IP messages of an inoperable ASP
- MERVA Link commands

Class-1 Journal Entries

Class-1 journal entries have the format of a MERVA Link PDU. This class comprises the events:

- Submit application message (SUBMIT.Request)
- Submit acknowledgment message (SUBMIT.Request)
- Deliver application message (DELIVER.Indication)
- Deliver acknowledgment message (DELIVER.Indication)

The events of this class are journaled only upon customization request. This customization option is specified separately for each ASP and for each of the three message categories (outgoing message, incoming report, and incoming application message).

MERVA Link guarantees that a message is not passed twice to the receiving application despite possible transmission failures. A message can, however, be submitted and transmitted twice. Therefore, a message can be recorded twice (or even more often) in the MERVA ESA journal.

Class-2 Journal Entries

Class-2 journal entries have the format of a message in a MERVA ESA queue buffer. This class contains the journal of an IP message of an inoperable or permanently waiting ASP. The message has been copied from the ACQ to the send queue of another message transmission medium using the MERVA Link **recover** command.

This class contains also the journal of an undeliverable message that was moved out of the ACQ to continue message transmission. It has been routed to an error queue either automatically or using the MERVA Link **iprecov** command. The events of this class are always journaled.

Class-3 Journal Entries

Class-3 journal entries have the format of a MERVA ESA command journal entry. The following MERVA Link commands are journaled:

- ACLOSE
- AOPEN
- ASTART (SA)
- HOLD (HA)
- KICKOFF (KA)
- LCRESET (LCRS)
- LRRESET (LRRS)
- RECOVER (IPCOPY)
- IPRECOV (IPMOVE)

The events of this class are always journaled when the command has been executed, successfully or not successfully. The MERVA ESA command, which is issued by a subset of these MERVA Link commands, is journaled separately by MERVA ESA. The command feedback is, however, in most cases the same in the MERVA ESA command journal entry and in the MERVA Link journal entry.

General Layout of a MERVA Link Journal Record

A MERVA ESA journal record written by MERVA Link consists of the following sequence of fields:

1. The 1-byte journal entry identifier (X'70' to X'7F').
2. The 24-byte journal record MERVA ESA key.
3. The 25-byte journal record user key.
4. The 4-byte journal record data length field in one of the two valid MERVA ESA length field formats (LLbb, or LLLL with the high-order bit on).
5. The variable length journal record data starting at displacement 54 (X'36') in a MERVA ESA journal record.

The last 7 bytes of the 24-byte journal record MERVA ESA key contain blanks unless the journal entry is segmented. A journal entry is automatically segmented in multiple journal records if the journal data is too large for a single record in the journal data set.

The journal entry identifier, the first 17 bytes of the 24-byte journal record MERVA ESA key, and the 25-byte journal record user key are the same in all journal records of a segmented journal entry.

Types of MERVA Link Entries in the MERVA ESA Journal

MERVA Link entries in the MERVA ESA journal are identified by a hexadecimal journal identifier starting with 7. The set of MERVA Link journal entries consists of the following members:

70	Outgoing application message
71	Outgoing acknowledgment message
72	Incoming application message
73	Incoming acknowledgment message
78	Recovered message
7F	MERVA Link command

All these journal entries are described in the following in detail.

Outgoing Application Message Journal Entry (ID=70)

An outgoing application message is journaled before it is passed to the message transfer system or to the first application support filter (if applicable) at the sending side. At this point in time it can have already been authenticated and encrypted. The journal record, however, shows plain text, not the encrypted text. Authentication and encryption control information is contained in the message heading (if applicable).

The journal identifier in the journal record header contains X'70'. The user extension of the journal record key contains the 4-byte MERVA Link MIP sequence number and the name of the applicable sending ASP.

The journal record data of an application message consists of the message heading and the body part data segment. The body part header is not contained in the journal record.

The 8-byte prefix of the body part data segment in a journal record does not conform to the rules of a P2 body part data segment. It consists of a 4-byte identifier X'00008122' or X'00008123', followed by a 4-byte field containing the length of the body part data including the 4-byte length field.

Outgoing Acknowledgment Message Journal Entry (ID=71)

An outgoing acknowledgment message (status report) is journaled before it is passed to the message transfer system or to the first application support filter (if applicable) at the sending side.

The journal record of a status report (acknowledgment message) consists of the journal record header and the status report data segment (SR-ASPDU).

The journal identifier in the journal record header contains X'71'. The user extension of the journal record key contains the 4-byte MERVA Link MIP sequence number and the name of the applicable sending ASP.

Incoming Application Message Journal Entry (ID=72)

An incoming application message is journaled before it is routed to the applicable incoming message queue(s). At this point in time, an application message text has already been authenticated and decrypted (if applicable). The message integrity has, however, not yet been checked according to the rules of the MERVA Link MIP.

The journal identifier in the journal record header contains X'72'. The user extension of the journal record key contains the 4-byte MERVA Link MIP sequence number and the name of the applicable sending ASP.

The journal record data of an application message consists of the message heading, and the message body part data segment. The body part header is not contained in the journal record.

The 8-byte prefix of the body part data segment in a journal record does not conform to the rules of a P2 body part data segment. It consists of a 4-byte identifier X'00008122' or X'00008123', followed by a 4-byte field containing the length of the body part data including the 4-byte length field.

Incoming Acknowledgment Message Journal Entry (ID=73)

An incoming acknowledgment message (status report) is journaled before it is correlated with the reported message in the ACK wait queue or before it is routed to the incoming report queue (without correlation with the reported message).

The journal identifier in the journal record header contains X'73'. The user extension of the journal record key contains the 4-byte MERVA Link MIP sequence number and the name of the applicable receiving ASP.

The journal record data of an acknowledgment message journal record contains a status report data segment (SR-ASPDU).

Delivery Notification Journal Entry (ID=74)

| A NOTIFY.Indication event is part of the asynchronous MERVA Link message
| transfer protocol. The support for the asynchronous message transfer service has
| been dropped, so the delivery notification journal entry is no longer applicable.

Recovered Message Journal Entry (ID=78)

The Recovered Message Journal Entry (ID=78) can contain either of two kinds of recovered messages. It can be an IP message that was copied to the send queue of another transmission medium, or an IP message that could not be successfully delivered to the receiving application and was moved out of the ACQ to continue message transmission.

An in-process message in the ACQ of an inoperable or permanently waiting ASP can be recovered (copied) using the MERVA System Control Facility command **recover**. This recover process is journaled immediately before the message is copied (routed as specified by the routing table associated with the ACQ, and kept as IP message in that ACQ).

At this point in time, the message class is set to RC, the PDM indicator is set, and the MERVA Link control fields SUBMIT return code (EKASUBRC), SUBMIT diagnostic code (EKASUBDC), SUBMIT date and time (EKASUBDT), MTL message identifier (EKAMSGID), MIP window size (EKAWSIZE), and MIP message sequence number (EKAAMSEQ) have been deleted in the copied message. The IAM message ID control field is not deleted.

The journal record of a copied IP message consists of the journal record header and the copied message in the MERVA ESA queue buffer format.

The journal identifier in the journal record header contains X'78'. The user extension of the journal record key contains the message class RC and the name of the applicable sending ASP. The 4-byte MERVA Link MIP sequence number of the in-process message is not contained in the copied message and not shown in the journal record header.

An ASP with an undelivered in-process message in the ACQ can be recovered using the MERVA System Control Facility command **iprecov**. This function can also be performed automatically upon ASP customization request by the ASP.

The recovered message is journaled immediately before it is routed as specified by the routing table associated with the ACQ, and deleted as IP message in the ACQ.

At this point in time, the message class has been set to RI, and the MERVA Link control fields window size (EKAWSIZE) and MIP message sequence number (EKAAMSEQ) have been deleted in the recovered message.

The journal record of a recovered message with class RI consists of the journal record header and the message in the MERVA ESA queue buffer format.

The journal identifier in the journal record header contains X'78'. The user extension of the journal record key contains the message class RI, the MERVA Link MIP message sequence number, and the name of the applicable sending ASP.

When an ASP is recovered from an undelivered message, the message following the undelivered message is assigned the same MIP message sequence number as the undelivered message to stay in a contiguous MSN sequence. This is why the MIP message sequence number in recovered messages in the MERVA ESA journal can be non-unique.

MERVA System Control Facility Command Journal Entry (ID=7F)

A MERVA System Control Facility command concerning MERVA Link that has an effect on the processing of messages by MERVA Link is journaled as soon as it has been executed (successfully or not successfully). At this point in time, the command response message returned to the operator for a subset of the journaled commands is available.

The journal record of a MERVA Link command has the same format as the journal record of a MERVA ESA command (ID=14).

The journal identifier in the journal record header contains X'7F'. The user extension of the journal record key is empty as in a MERVA ESA command journal entry. The journal record data area contains the operator identification, originator information as specified in the MERVA ESA user record of this operator, the command with its parameter, and the command response message (if applicable).

Part 3. MERVA Link USS Functions

Chapter 9. The MERVA Link USS Control Facility

The MERVA Link USS Control Facility is the collection of resources used to establish the MERVA Link USS environment, and to control the execution of MERVA Link processes, both manually and automatically. The MERVA Link USS Control Facility consists of the MERVA Link application control table (ACT), the MERVA Link application control daemon (ACD), the MERVA Link configuration and security files, and a set of control applications.

Control Facility Overview

The components of the MERVA Link USS Control Facility are described in the following sections. All resources related to ASPs are defined in the MERVA Link USS structures, and are partly supported by the MERVA Link USS programs. However, in a MERVA Link USS Gateway environment, ASP resources are used for test purposes only.

Application Control Table (ACT)

The MERVA Link USS application control table (ACT) is the most important MERVA Link USS resource. It contains static MERVA Link customization parameters as well as dynamic processing information. Any MERVA Link process must attach the ACT during its initialization, and detach the ACT before termination. A process that is attached to the ACT can retrieve information from the ACT and store dynamic status information in the ACT.

The ACT is divided into these sections:

- ACT header
- ACT ASP section
- ACT ISC section

For more information about the structure of the ACT and the information contained in the ACT, see "Application Control Table (ACT)" on page 97.

Application Control Daemon (ACD)

The MERVA Link USS application control daemon (ACD) is a long lasting OS/390 USS process (daemon) that represents a MERVA Link USS instance. The ACD can be started as part of the OS/390 USS system startup (/etc/inittab), by an authorized OS/390 USS user, or by other means. The termination of the ACD can be requested by a MERVA Link application control command or by OS/390 USS shell commands.

The main tasks of the ACD are:

- Allocate shared memory for the application control table (ACT)
- Initialize the ACT with static MERVA Link customization parameters from the applicable configuration file
- Copy confidential information (passwords) from security files in a confidential way into ACT ISC entries
- Check for and handle an ACD termination request

For more information about the ACD, see "Application Control Daemon (ACD)" on page 101.

Application Control Command Application (ACC)

The MERVA Link USS application control command application (ACC) can be used by any authorized user to display MERVA Link resources and to modify MERVA Link processing parameters in a USS command shell environment. Examples of ACC functions are:

- Display ACC help information
- Display ACT header information
- Display a list of the partner nodes (ISC parameter sets) defined in the ACT
- Display static and dynamic information of an intersystem connection to a specific partner node
- Analyze error information in the ACT, and explain it
- Request ACD termination

The requested ACC function is entered as the first parameter of the USS command **ekaacc**. This parameter is called the ACC command name. Additional command parameters may be applicable.

To get a summary of the ACC commands, enter **ekaacc** without a parameter.

For more information about the ACC application, see "Application Control Command Application (ACC)" on page 103.

Configuration and Security Files (CFG, SEC)

The static information used by the ACD to initialize an ACT is contained in a MERVA Link USS Configuration File (CFG). The full path name of the configuration file must be specified as a command parameter when the ACD is started. CFG information types and structures are the same as the types and structures of the information in the ACT.

An application control function is provided to generate a configuration file based on information in the currently active ACT. An exported configuration file shows the keywords for all parameters supported by the MERVA Link USS configuration facility.

Conversation security information to access partner systems via TCP/IP is stored in MERVA Link USS Security Files (SEC). There is one security file for each partner system that must be connected via TCP/IP.

Conversation security information in a security file is encrypted. It can be decrypted only in the original host system by an authorized process.

Conversation Security Programs (ACS and CSI)

There are two MERVA Link USS programs that provide a USS shell command interface to store confidential conversation security information:

- The local security control application (ACS) lets you store information in MERVA Link USS security files when the ACD is active. For more information about the ACS, see the *MERVA for ESA Customization Guide*.

- The partner security control application (CSI) lets you store security information at a partner MERVA Link system. For more information about the CSI, see "Partner Security Control Application (CSI)" on page 112.

Configuration Verification Programs (VCD, VCC, and VCS)

The MERVA Link configuration verification programs are:

- The application control daemon for verification (VCD), which is similar to the ACD. Like the ACD, it lets you verify a configuration file and for generating an ACT, but for configuration verification purposes only. In a configuration verification environment, use the VCD as you would use the ACD in a regular environment. For more information about the VCD, see "Application Control Daemon for Verification (VCD)" on page 114.
- The application control command application for verification (VCC), which is similar to the ACC. It lets you display information in the ACT owned by a VCD, and terminate a VCD. In a configuration verification environment, use the VCC and EKAVCC as you would use the ACC and EKAACC in a regular environment. For more information about the VCC, see "Application Control Command Application for Verification (VCC)" on page 116.
- The local security application for verification (VCS), which is similar toe th ACS. It provides a USS shell command interface that lets you store confidential information in MERVA Link USS security files when the VCD is active. For more information about the VCS, see "Local Security Control Application for Verification (VCS)" on page 118.

Application Control Table (ACT)

The MERVA Link USS application control table (ACT) is the most important MERVA Link USS resource. It is accessed and shared by all MERVA Link USS processes. The ACT provides static and dynamic information to processes, and provides for communication between MERVA Link USS processes. For example, a MERVA Link USS routing process can retrieve from the ACT information needed to connect to a partner system, and can store process statistics and error information in the ACT.

The configuration of a MERVA Link USS instance is subject to modification when the MERVA Link network changes. To provide for an uninterrupted MERVA Link USS service, a dual ACT facility is provided by MERVA Link USS. Dual means that two ACTs, an old and a new ACT, may exist for a short period of time when the configuration of a MERVA Link USS instance is changed. The old ACT is used by processes that were already active when the new ACT was generated. The old processes continue to use the old ACT until they end. When all old processes have ended, the old ACT is physically removed. The new ACT is used by all new processes.

An ACT is generated or an ACT configuration is changed simply by starting the MERVA Link application control daemon (ACD) with the applicable configuration file, or starting a second ACD instance with a modified configuration file. Another configuration change can be performed by starting another ACD instance when the former old ACT has been physically removed.

ACT IPC Resources

The ACT is implemented as an OS/390 USS shared memory region (SHM) for read/write access by all MERVA Link processes. The MERVA Link application control daemon (ACD) allocates the SHM for the ACT, and initializes the ACT with static information contained in a MERVA Link USS Configuration File. The ACD releases the SHM of the ACT when it is terminated.

An OS/390 USS SHM is associated with an IPC resource, which is an HFS file. Any MERVA Link ACT is therefore associated with an IPC resource. The name of the IPC resource must be specified when the SHM is created, and when a process attaches to the SHM.

The HFS files that are used as IPC resources for the dual ACT facility are contained in the MERVA USS IPC directory. The file names are **ekaact.a** and **ekaact.b**. These files can be generated by a **touch** command in an OS/390 USS shell. An IPC resource file can be empty.

ACT Header

The ACT header contains the local MERVA Link node name as the only static MERVA Link customization parameter. All other information in the ACT header is either dynamic information, or static information that cannot be customized.

ACT Identifiers

The ACT header starts with three ACT identifiers:

- An ACT version identifier (1)
- The suffix of the corresponding IPC resource file (a or b)
- The ACT name

The ACT name field contains the name of the ACT in uppercase letters: **EKAACT**. The application control daemon (ACD) stores this character string in the ACT after it has been sucessfully initialized. The ACD can modify this string to indicate that the ACT is not accessible. In fact, the string is modified to **eKAACT** before the ACD terminates to indicate a request for immediate termination of any MERVA Link process.

Any process that accesses the ACT must check this field. If it does not contain **EKAACT**, the ACT must be considered inaccessible, and the process must terminate as soon as possible.

ACT Generation Time

The ACT generation time field contains a timestamp that reflects the date and time when the ACT was initialized. This timestamp is used to select the new ACT if two ACT SHMs are allocated.

Local MERVA Link Node Name

The local MERVA Link node name field contains the node name of the MERVA Link USS Gateway, the local MERVA Link system. This name must not be specified as destination node by ASPs in partner MERVA Link systems as long as Application Support (AS) functions are missing from MERVA Link USS.

Inbound messages that specify this node name as the recipient node are accepted, and passed to a dummy inbound AS program (EKAASI). This dummy AS program prints inbound message information to a processing trace file. The MERVA Link USS Gateway does not support ASPs that deliver inbound messages to a MERVA ESA message queue.

Inbound messages that do not specify this node name as the recipient node are routed to the specified node or rejected as undeliverable (depending on the availability of routing information for the specified recipient node).

ACD Process Identifier

The ACD process identifier field is the first field of the ACD section in the ACT header. The ACD process identifier is saved in the ACT header to enable any program that is attached to the ACT to send an alarm signal to the MERVA Link daemon. An alarm signal wakes the MERVA Link daemon up if it is sleeping.

ACD Process Owner Identifier

The ACD process owner identifier is saved in the ACT header to control the execution of restricted commands. Restricted commands, for example ACD termination, are accepted only from the ACD process owner, or an operator with root user authority (UID 0).

ACD Sleep Time Interval

The ACD sleep time interval specifies the number of seconds the MERVA Link daemon must sleep between phases of activity. This time interval applies only if the daemon ist not activated by any other event during a sleep time interval.

ACD Activity Trace Area

The MERVA Link daemon writes operator messages to the ACD activity trace area in the ACT header. The ACD activity trace area can contain up to 16 operator messages of up to 69 characters each. The MERVA Link ACC command **dsd** can be used to display the ACD activity trace.

Processing Trace Parameters

Information concerning the MERVA Link processing trace:

- Trace level
- Trace wrap-around limit
- Trace file index for an inbound SNA APPC conversation
- Trace file index for an inbound TCP/IP conversation
- Trace directory path name (applies to all inbound and outbound MERVA Link processes)

These parameters are described in “Chapter 13. MERVA Link USS Problem Determination Aids” on page 149.

Unsolicited Receiving Process Status Information

An error can be found in a MERVA Link receiving process before the identity of the receiving process (ASP) or the routing destination was determined. Information describing this kind of error is stored by one of the MERVA Link programs EKATPI, EKATCI, and EKAP1I in the applicable fields of the ACT header. The status information stored by these programs consists of:

- For EKATPI, the status timestamp, the EKATPI return and reason codes, and the USS error number (if any)
- For EKATCI, the status timestamp, the EKATCI return and reason codes, and the USS error number (if any)
- For EKAP1I, the status timestamp and the EKAP1I return and reason codes

ACT ASP Section

The ACT ASP section consists of a sequence of ACT ASP entries. A pointer to the first ASP entry and the total number of ASP entries is contained in the ACT header.

ACT ASP entries are not applicable in a MERVA Link USS Gateway scenario. They are supported by MERVA Link USS for test purposes only.

ACT ISC Section

The ACT ISC section consists of a sequence of ACT ISC entries. A pointer to the first ISC entry and the total number of ISC entries is contained in the ACT header. The fields of an ACT ISC entry are described in the following.

Partner Node Name

The partner node name can be considered as the key of an ISC entry. The ISC entry contains intersystem connection parameters that apply for a connection to a specific partner node. The ISC parameters either describe the partner system that houses the partner node, or describe a gateway that must route the conversation to one of its partner systems (that houses the partner node).

A routing process uses the inbound destination node name to find the applicable ISC parameter set to perform its task.

SNA APPC Connection Parameters

The SNA APPC connection parameters are the name of an APPC/MVS Side Information (SI) Profile, that is also called a symbolic destination, and an optional partner TP name if the TP name in the SI profile must not be used.

An APPC/MVS SI profile contains the following parameters:

- Partner LU name
- SNA mode name
- Partner TP name

An SNA APPC connection is not supported if a symbolic destination is missing from an ACT ISC entry.

TCP/IP Connection Parameters

TCP/IP connection parameter fields are defined in the ACT ISC entry structure for both of the following:

- Partner host name
- Partner port number

A TCP/IP connection is not supported if a partner host name is missing from an ACT ISC entry.

Conversation Security Parameters

Conversation security parameter fields are defined in the ACT ISC entry structure. These fields are:

- Conversation security user ID
- Conversation security user password (encrypted)
- Local password encryption method
- Partner password encryption method

Conversation security information applies only for outbound TCP/IP conversations. In the MERVA Link USS SNA APPC environment, conversation security is completely handled by APPC/MVS.

ISC Status Information

An error can be found in a MERVA Link sending or receiving TP when the identity of the partner node is known. Information describing this kind of error is stored by the MERVA Link programs EKATPO, EKATCO, EKATPI, and EKATCI in the applicable fields of the ACT ISC entry.

The status information for each of the four TPs consists of the status timestamp, the TP return and reason codes, and the USS error number. Separate sets of status information fields are defined for processes handling inbound and outbound messages (currently not applicable), and for routing processes. The set of routing process status information fields includes EKAP1R return and reason code fields.

Extended Error Information

Extended error information that has been found in an error report received from the subject partner system is saved in the corresponding ACT ISC entry for access by other functions. It can be:

- Diagnostic code originator type
- Error code vector
- Error code vector receive timestamp

ISC Statistics Information

The following statistics information is collected by a routing process, and stored in the ACT ISC entry describing the destination node of the routing process:

- Number of messages routed since the ACT generation
- Amount of time used to route messages (in seconds)

The routing process statistics apply to all processes with the subject node as destination node, independent of the originating node.

Event Delay Information

Delay information is collected by a routing process for the following events:

- Connect Request
- Send Request
- Request Confirmation
- Receive Request

The following information is collected for each event, and stored in the applicable ACT ISC entry:

- Minimum event delay
- Average event delay
- Maximum event delay
- Number of measured events
- Accumulated event delay

The average delay is always the accumulated event delay divided by the number of measured events.

Application Control Daemon (ACD)

The MERVA Link USS application control daemon (ACD) is a long-lasting background process that generates and owns the MERVA Link USS ACT. In the MERVA Link USS Gateway environment it has nothing else to do but waiting for a request to terminate.

The ACD must be active and must have generated an ACT before any other MERVA Link process can execute. An active ACD and its ACT represent a particular MERVA Link USS instance. There is one ACD for each MERVA USS instance in a host.

All aspects of the ACD are described in the following sections.

Starting the ACD

The ACD is started by an authorized USS user (the MERVA USS instance owner) via a USS shell command, via an OS/390 batch job, or by other means. It is started with a set of parameters:

- The name of the MERVA USS instance directory must be specified as the first ACD command parameter. It identifies the applicable MERVA USS instance.
- The full path name of the configuration file must be specified as another ACD command parameter. The configuration file is the source of the MERVA Link USS customization.
- To request an external trace from the ACD, specify the full path name of a trace directory. If no trace directory is specified, the ACD does not generate an external trace.

External ACD Activity Trace

The ACD supports an external activity trace. The activity trace is requested by specifying a trace parameter as an ACD command line argument. An ACD trace parameter consists of the keyword **trc** that is optionally followed by the applicable trace directory name. An external activity trace is not written by default.

The default trace directory is the subdirectory **trc** of the MERVA USS instance directory (for example, /u/merva1/trc/) if a trace directory name is missing from the ACD trace parameter.

The activity trace of the ACD is written to the file **ekaacd.t.MMDDhhmmss** in the applicable trace directory. The user who starts the ACD must be authorized to write files to the trace directory. Otherwise, the activity trace is not written. There is no error indication in the latter situation.

Internal ACD Activity Trace

An activity trace area is defined in the ACT header for use by the ACD. It contains 16 entries. An ACD activity trace entry consists of a timestamp and an activity message of up to 69 characters.

The activity trace area is used by the ACD in wrap-around mode. This means, the 17th activity trace message overlays the first activity trace message. The second trace entry is cleared in this example to indicate the wrap-around situation.

The ACD writes an activity trace message for all events it considers as interesting enough to be reported. Interesting events are, for example:

- Automatic wake-up
- Start of ACD termination

The ACD activity trace messages are self-explanatory. They are not further explained.

The activity trace messages written to the ACT header are also written to the external ACD trace (if applicable).

Monitoring ACD Activity

The MERVA Link application control command application (ACC) provides a command that can be used to display the current conntent of the ACD activity trace area in the ACT header. The command name is **dsd** (for **d**isplay the **s**tatus of the application control **d**aemon). This command has no parameters. The output of the **dsd** command consists of a heading line and up to 16 ACD activity trace lines.

The heading line of the **dsd** command output shows the USS process identifier of the ACD and the time when the ACT was last active.

Each ACD activity trace line starts with the time when the activity trace was generated. The time is followed by an ACD activity trace message of up to 69 characters. A maximum of 16 trace lines are supported by the ACD and displayed the **dsd** command. A trace area wrap is indicated by a separator line following the most recently written activity trace line.

Stopping the ACD

The termination of the ACD can be requested via the MERVA Link application control command **trm daemon**, or by sending an interrupt signal (SIGINT) to the ACD process. The ACC command to terminate the ACD is accepted only from the ACD process owner and from a USS root user with uid=0.

Application Control Command Application (ACC)

The MERVA Link USS application control command application (ACC) is the means by which you control the status and the execution of MERVA Link USS. ACC can be used in an OS/390 USS interactive command shell environment, in a USS batch environment (BPXBATCH), and in an MVS batch environment. The command or program **ekaacc** calls ACC in the OS/390 USS environments. Program **EKAACC** calls ACC in the MVS environment.

An ACC interactive command begins with an ACC command name that must be entered in lowercase letters. A subset of the ACC command names is followed by a resource name. Resource names can be, for example, an ASP name, a partner system node name, a keyword, or a MERVA Link diagnostic code. MERVA Link resource names are normally made of uppercase characters. They can, however, be entered with lowercase letters. An uppercase translation is performed automatically (where applicable). Help information is displayed as response to an invalid ACC command.

A conversation mode is supported by ACC. The USS command **ekaacc sc** starts the ACC conversation mode in a window. All data entered in a window in ACC conversation mode is interpreted as an ACC command. The command prompt **ekaacc** is a reminder of the fact that the window is in ACC conversation mode.

A batch input mode is supported by ACC. Use the program calls **ekaacc si** (BPXBATCH) or **EKAACC si** (MVS) to start the ACC in batch input mode. The ACC commands are retrieved from **stdin**, and the command output is written to **stdout**.

ACC cannot be used when the ACD is not active. The ACD generates and owns the ACT, which is the main resource of the MERVA Link USS Control Facility.

Note: The support of ASP-related functions has not been removed from the ACC implementation in the OS/390 USS environment. ASPs are, however, supported by the MERVA Link USS Gateway for installation verification and test purposes only. All functions related to ASPs are therefore not applicable in a production environment.

ACC Command Format

An ACC command consists of a command name and a command parameter. A command name consists of up to four characters. The command parameter can be the name of a resource or the resource itself. A resource (for example, a diagnostic code) can consist of two parts. The command parameter is empty in a subset of the ACC commands.

The majority of the ACC command names is structured according to the following rules:

- The first letter identifies a command activity, for example, **a**nalyze, **c**hange, **d**isplay, **h**elp, **l**ist, send **p**robe, **r**eset, **s**et, **s**tart, and **t**erminate.
- A command activity can also be identified by the first two characters of a command name, for example, **cx** (configuration export) and **dx** (display explanation).
- The character behind the activity indicator identifies a data class if it is not the last character of the command name. Data classes are, for example, **e**rror code vector, **p**arameter, **r**eceiving process, **r**outing process, **s**ending process, **s**tatus information, and **t**race information.
- The last letter of the command name identifies an object or object class, for example, **a** (ACT ASP entry), **c** (ACT ISC entry), **h** (ACT header), **d** (diagnostic code), and **e** (error information or error class).

Most of the ACC commands are described in Chapter 8 of the *MERVA for ESA Operations Guide*. The advanced ACC commands that are not described there are descirbed in the following in alphabetic sequence of the ACC command name.

Changing ACT ISC Parameters for Partner Nodes

The following commands let you modify parameters in ACT ISC entries for partner nodes temporarily, that is, during the lifetime of the current ACT shared memory instance. The modifications are lost when the ACD is terminated.

Change partner symbolic destination name

To change the SNA APPC symbolic destination name in the ACT ISC entry for a partner node, enter

cdc ***p_node sdn***

where ***p_node*** is the name of the partner node, and ***sdn*** is the new symbolic destination name. The new symbolic destination name is not checked against APPC/MVS side information definitions.

Change partner host name

To change the TCP/IP partner host name in the ACT ISC entry for a partner node, enter

chc ***p_node host***

where ***p_node*** is the name of the partner node, and ***host*** is the new partner host name. The new partner host name is checked against the TCP/IP customization, and must be a valid TCP/IP host name defined in the local USS system or in a remote TCP/IP name server.

Change partner host port number

To change the TCP/IP port number in the ACT ISC entry for a partner node, enter

cpc ***p_node port***

where ***p_node*** is the name of the partner node, and ***port*** is the new port number. The new port number must be a number from 1000 to 32000.

Change partner password encryption method
To change the partner password encryption method in the ACT ISC entry for a partner node, enter

cpe *p_node methodnum*

where ***p_node*** is the name of the partner node, and ***methodnum*** is the identifier of the encryption method:

0	MERVA Link basic encryption method.
1	An encryption method using the unrestricted DES function **crypt()**. This method can be used only for a connection to another MERVA Link USS node.

Change partner TP name
To change the SNA APPC partner TP name in the ACT ISC entry for a partner node, enter

ctc *p_node tpn*

where ***p_node*** is the name of the partner node, and ***tpn*** is the new partner TP name. This TP name overwrites the TP name specified in the APPC/MVS side information.

Displaying Information about Routing Process Event Delays

A MERVA Link USS routing process collects information about the delays associated with the following event categories:

- Connect to destination partner node, including probe confirmation
- Send PDU to destination partner node
- Get message window confirmation from destination partner node
- Receive next PDU from orginator partner node

The delays are measured at the MERVA Link P1 layer close to the corresponding service primitive calls. The processing time of the P1 functions are not included in the delay values.

The ACC command **ddc partner_node_name** displays event delay information collected for the specified partner node. This information consists of the number of events, and the minimum, average, and maximum delay values for each event category. The delay values are displayed in milliseconds.

The send delay can be a very small value. It is the time interval required by the data communication subsystem (APPC/MVS or TCP/IP) to copy the message PDU into its buffers. The actual transmission to the partner system may not be included in this time interval.

The receive delay can be a very large value. It is the time interval until the next message PDU is received from the originating partner system. This time interval may include the time required by the originating node to handle the confirmation of a message window. It can be 3000 milliseconds for a window size of 100 messages.

The ACC command **ddcr partner_node_name** displays event delay information collected for the specified partner node, and resets all event delay information in the ACT ISC entry.

Modifying an ASP

The MERVA Link ASP operating commands let you modify the processing characteristics of a sending or receiving ASP. ASP operating commands are of two types:

- Local ASP commands, which modify the processing characteristics of local ASPs (these commands start with the letter **m**)
- Partner ASP commands, which modify the processing characteristics of partner ASPs (these commands start with the letter **p**)

The second letter in the command name identifies the specific function to be performed. You always specify the name of the local ASP as the command parameter, even when modifying the processing characteristics of its partner ASP.

To carry out an ASP operating command for a partner ASP, MERVA Link uses command probes, which it sends to the partner system. MERVA Link command probes are not supported by all MERVA Link implementations. A MERVA Link implementation that does not support command probes handles an inbound command probe as if it were a T-Probe.

Disable local receiving ASP
The command **mda** ***local_asp_name*** requests that the specified receiving ASP is disabled. A disabled ASP does not accept any inbound message.

Enable local receiving ASP
The command **mea** ***local_asp_name*** requests that the specified receiving ASP is enabled. An enabled ASP accepts inbound messages from its partner ASP.

Hold local sending ASP
The command **mha** ***local_asp_name*** requests that the specified sending ASP is set to HOLD status. An ASP in HOLD status does not get messages from the send queue cluster and transfer those messages. It can, however, transfer a message in the MERVA Link control queue.

Kick off local sending ASP
The command **mka** ***local_asp_name*** requests that the specified sending ASP is kicked off. An ASP in HOLD status keeps this status when it is kicked off.

The kick-off command asks the MERVA AIX daemon to send an alarm signal to the MERVA Link daemon to request immediate processing of the kickoff request. Sending a signal to the MERVA daemon can fail if you are not authorized to do so. Your kick-off request is handled in this case when the MERVA Link daemon is next activated by any other means. Your kick-off request is not lost.

Start local sending ASP
The command **msa** ***local_asp_name*** requests that the specified sending ASP is set to NOHOLD status and be kicked off.

Disable partner receiving ASP
The command **pda** ***local_asp_name*** requests that the partner receiving ASP of the specified local ASP is disabled. A disabled ASP does not accept any inbound message.

Enable partner receiving ASP
The command **pea** ***local_asp_name*** requests that the partner receiving ASP of the specified local ASP is enabled. An enabled ASP accepts inbound messages from its partner ASP.

Hold partner sending ASP
The command **pha** ***local_asp_name*** requests that the partner sending ASP of the specified local ASP is set to HOLD status. An ASP in HOLD status does not get messages from the send queue cluster and transfer those messages.

Kickoff partner sending ASP
The command **pka** ***local_asp_name*** requests that the partner sending ASP of the specified local ASP is kicked off. An ASP in HOLD status keeps this status when it is kicked off.

Start partner sending ASP
The command **psa** ***local_asp_name*** requests that the partner sending ASP of the specified local ASP is set to NOHOLD status and be kicked off. An ASP in NOHOLD status can get messages from the send queue cluster and transfer those messages.

Send Probe Commands

The **send probe** commands provide a means to send a PROBE to the partner ASP of a local ASP. The name of the local ASP must be specified as probe command parameter. Standard MERVA Link MT layer programs (ekap1o, ekatpo, ekatco) are used to handle send probe commands.

Standard MERVA Link AS layer programs (ekaaso, ekap2o) are not involved in an ACC send probe command. The ACC command processor itself represents the AS layer programs. This is why the ACC command processor stores return information from the connect request associated with the send probe command in the ASO and P2O return information fields in the ACT. These fields are the ASO status and diagnostic codes, the ASO return and reason codes, and the P2O return and reason codes.

If an ACC send probe command was successfully executed and the connect request was confirmed by the partner ASP, the ASP status and diagnostic codes are set to 00 CON CF. Any error is reported as appropriate.

Send T-Probe
The command **pta** ***local_asp_name*** requests that a T-Probe be sent to the partner ASP of the local ASP specified. When the command has been executed, you are informed whether the ASP availability test was successful or unsuccessful.

To display the processing trace supported by the MERVA Link programs handling the send probe command, specify **pta1** ***local_asp_name*** (for trace level 1) or **pta2** ***local_asp_name*** (for trace level 2). The processing trace shows error information when the probe is not successful.

Send R-Probe

The command **pra** ***local_asp_name*** requests that an R-Probe be sent to the partner ASP of the local ASP specified. The interface of the **pra** command is the same as the **pta** command interface. The difference is the function performed by the R-Probe.

In addition to the function performed by a T-Probe, the R-Probe tests the connection from the partner system back to the local system. This test is, however, performed only if all the following are true:

- The receiving partner ASP is available.
- The sending ASP at the partner system is currently not active.

- The partner MERVA Link system can handle an inbound R-Probe. MERVA Link ESA, for example, does not support an R-Probe, and handles an R-Probe as if it were a T-Probe.

Analyzing and Explaining Error Information

The error codes collected during a MERVA Link process are saved in the ACT header, in an ACT ASP entry, and in one or two ACT ISC entries.

- The **display status** commands (**dsh**, **dsa**, and **dsc**) show the complete error information stored in an ACT object (ACT header, ACT ASP entry, ACT ISC entry). That information is not necessarily related to a single instance of a MERVA Link application process (send a message, receive a message, or route an inbound conversation), nor is it always the complete information associated with an instance of a MERVA Link process.
- The **analyze** commands (**asa**, **aih**, **ara**, and **arc**) divide the error information in the ACT objects into error analysis process groups, and explain the error information from an application process perspective. The MERVA Link error analysis process groups are:

  Send Message Process
  The send message process includes all activities related to an outbound conversation initiated by a sending ASP.

  Accept Inbound Conversation Process
  This process is the begin of any inbound conversation. It becomes a receive message process when the intended recipient of the message is a local receiving ASP. It becomes a route inbound conversation process when the recipient node is not the local node, and connection information for the recipient node is available in an ACT ISC entry.

  The error codes of an accept inbound conversation process are stored in the ACT header because this process is not yet associated with any ASP or ISC entry in the ACT. When the recipient node of an inbound message is not the local node, and no ACT ISC entry is available for the recipient node, an error is reported (by the inbound P1 processor). This error information is also stored in the ACT header.

  Receive Message Process
  This process includes the activities of the inbound P1 and P2 processors, and the activities of the inbound AS processor. Errors of the receiving TP (EKATPI or EKATCI) are not analyzed as part of a receive message process.

  Route Inbound Conversation Process
  This process includes the activities of the inbound P1 processor that performs the P1 routing function, and the activities of the sending TP (EKATPO or EKATCO) that passes the inbound conversation to the destination node. Errors of the receiving TP (EKATPI or EKATCI) are not analyzed as part of a route inbound conversation process.

ACC Conversation Mode

A sequence of ACC commands can be entered as a sequence of independent USS commands calling the MERVA Link program **ekaacc**. As an alternative, the MERVA Link program **ekaacc** can be called and put in ACC conversation mode. Any terminal input is then interpreted as an ACC command.

Start ACC Conversation Mode

The USS command **ekaacc sc** places the terminal in ACC conversation mode. You are informed about that fact in the first operator message that is displayed as response to the command **ekaacc sc**. The second operator message tells the applicable MERVA instance. The third operator message tells how to exit the ACC conversation mode.

ACC Command Echo

After typing an ACC command (or pressing the ENTER key without an ACC command), the ACC command that will be passed to the ACC command processor is displayed by **EKAACC** followed by the effective ACC command.

Effective ACC Command

At the begin of an ACC conversation the previous ACC command is set to **?**, an invalid command. A summary of the EKAACC commands is therefore displayed if you press the ENTER key without any data at the begin of an ACC conversation.

The ACC command that becomes effective after pressing the ENTER key in ACC conversation mode depends on the previous ACC command and on the data entered before pressing the ENTER key. The following rules apply:

- If the ENTER key is pressed without any data, the previous command is executed again.
- If data is entered before pressing the ENTER key, the new effective command is the entered data (explicit command) or it is based on this data (generic command).
- For a subset of the ACC commands, the first parameter of the previous command is automatically included in the new command if a mandatory command parameter is missing. This command subset contains all commands that require an ASP or partner node name as command parameter.
- The command name and parameters of the previous command can be included in the new command upon request. The command data can refer to parts of the previous command using a period (.) as the first character of the first and the second data token.

A Way to Make It Easier to Enter ACC Commands

When entering an ACC command, you can use a period (.) to refer to the previous command name or to the previous command parameter. For example:

- To display the customization parameters of the partner node USSNODE1, enter **dpc ussnode1**. To then display the status information contained in the ACT ISC entry for that node and to get an explanation of the error information in that ISC entry, enter **dsc ussnode1** or, to save effort, **dsc .**. The ACC interprets the period to mean ″the argument specified in the previous command″.
- To display an explanation of the first two error codes of the MERVA Link ESA sending MTP error code vector 000a0022ac020857, enter **dxsm 000a0022**. After entering this command, to display an explanation of the third error code as well, enter **dxsm 000a0022ac02** or, to save effort, **. .ac02** (note that a blank separates the two periods). The ACC interprets the first period to mean ″the previous command″ and the second period to mean ″the previous argument″. Next, enter **. .0857**; this is equivalent to entering **dxsm 000a0022ac020857**, and displays an explanation of the complete error code vector.

ACC Command Execution

ACC in conversation mode is represented by one long lasting process (ACC parent process) that is associated with a terminal. It is also associated with the MERVA

Link USS ACT when it starts. The ACT is detached when the ACC conversation mode is entered. The ACC parent process ends when ACC conversation mode end is explicitly requested.

An ACC command entered in ACC conversation mode is executed in an ACC process of its own (ACC child process). The ACC parent process forks an ACC child process to execute the subject command. The command output is written to stdout, and the ACC child process ends.

This technique is worthwhile to know in order to understand the behaviour of ACC when the ACT is refreshed while an ACC process is active in conversation mode.

End ACC Conversation Mode

The ACC commands **x** and **end** end the interactive ACC parent process. The Ctrl_C key that generates an interrupt signal (SIGINT) for the ACC process can be used in an rlogin shell to end ACC.

ACC conversation mode end must be explicitly requested. The ACC parent process does not end automatically when the ACD has been terminated.

ACC Batch Input Mode

When ACC is started in batch input mode, it reads a sequence of ACC commands (without the program name ekaacc) from stdin, and executes the commands sequentially in a single process. The command output is written to stdout.

Start ACC Batch Input Mode

The USS command **ekaacc si** starts program ekaacc in ACC batch input mode. The MERVA USS instance directory name can be optionally provided as the **si** command parameter, for example, **ekaacc si /u/merva1/**.

ACC does not prompt for commands in batch input mode.

ACC Command Execution

All ACC commands read from stdin in ACC batch input mode are executed in the ACC process that reads the commands from stdin. This is the major difference between ACC in conversation mode and ACC in batch input mode.

ACC in batch input mode is represented by a single process that is associated with the standard input and output files, and with the MERVA Link USS ACT that is applicable when the ACC process starts. This process ends when all ACC commands on stdin have been executed. An ACT refresh while an ACC process is active in batch input mode has no effect on that ACC process. It remains attached to the old ACT.

End ACC Batch Input Mode

The ACC commands **x** and **end** end the ACC batch input mode. ACC batch input mode ends also if EOF is found on stdin.

Execution Environments of ACC in Batch Input Mode

Three environments are supported by ACC in batch input mode:

- OS/390 USS shell environment
- OS/390 USS batch environment (BPXBATCH)
- OS/390 batch environment (EKAACC)

The following is an example of a command to execute ACC in batch input mode in a USS shell:

```
ekaacc si < acc.cmd > acc.out
```

In this example, the ACC commands are retrieved from file acc.cmd, and the ACC command output is written to file acc.out.

ACC Batch Job Samples

The following is an example of a job to execute ACC in batch input mode in a USS batch environment:

```
-------------------------------------------------------------------
//xxxxxxx  JOB xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
//EXEC  EXEC PGM=BPXBATCH,
//           PARM='PGM /usr/lpp/merva1/bin/ekaacc si'
//STDOUT  DD PATH='/u/user/acc.out',
//           PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//           PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//STDIN   DD PATH='/u/user/acc.cmd',PATHOPTS=ORDONLY
//STDENV  DD *
MERVA_DIR=/u/merva1/
/*
//
-------------------------------------------------------------------
```

The ACC commands are retrieved from file /u/user/acc.cmd, and the ACC command output is written to file /u/user/acc.out, in this example. The MERVA instance directory environment variable must be provided in STDENV.

The following is an example of a job to execute ACC in batch input mode in an MVS batch environment:

```
-------------------------------------------------------------------
//xxxxxxxx JOB xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
//EKAACC   EXEC PGM=EKAACC,REGION=4M,PARM='/si /u/merva1/'
//STEPLIB  DD DSN=hlq.SDSLLIB0,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
dph
lc
x
/*
//
-------------------------------------------------------------------
```

The ACC commands are retrieved from instream data SYSIN, and the ACC command output is written to SYSOUT, in this example. The MERVA instance directory name must be provided as another ekaacc command parameter when ACC is started in batch input mode in the MVS batch environment.

The STEPLIB refers to an OS/390 program object library (PDSE) that contains the MERVA Link USS programs and dynamic load libraries (DLLs). Replace **hlq.SDSLLIB0** by the name of this PDSE in your installation.

For more information about the generation of the MERVA Link USS program object library and about executing MERVA Link USS programs in the OS/390 MVS environment, refer to “Chapter 10. MERVA Link USS in the OS/390 USS Environment” on page 119.

Partner Security Control Application (CSI)

The MERVA Link USS partner security control application (CSI) provides a means to specify and verify conversation security information in the local system, transmit it to a partner system, and store it in the configuration facility of a partner MERVA Link system. The CSI can be used only for partner MERVA Link systems that support the CSI application; for example, MERVA Link USS, MERVA Link AIX, and MERVA Link NT. The CSI application is not supported by MERVA Link ESA.

The MERVA Link USS applications ACS and CSI are both used to specify and store conversation security information for use by MERVA Link processes:

- ACS handles security information for use by a local sending process.
- CSI handles security information for use by a sending process in a partner MERVA Link system.

ACS and CSI are MERVA Link USS applications with many common characteristics, for example, command parameters and the execution environments. The ACS is described in the *MERVA for ESA Customization Guide*. The CSI characteristics that are different from ACS are described in the following sections.

The CSI Program

The CSI program name is **ekacsi**. In an OS/390 USS shell environment, the command **ekacsi** is used to call CSI. The CSI can run in an OS/390 TSO USS shell or in a remote login shell at a remote host.

In an OS/390 batch environment, the program name **EKACSI** is used to start the CSI.

The CSI Execution Environment

The CSI process must have access to an active MERVA Link USS ACT. This is why the ACD must be active when the CSI is called, and the name of the MERVA USS instance directory must be accessible to the CSI program.

The fully qualified path name of the MERVA USS instance directory can be provided as the first CSI command parameter. If this parameter is missing when the CSI is called in a USS shell, the USS shell environment variable **MERVA_DIR** must contain the MERVA USS instance directory name. If this environment information is missing or incorrect, the CSI cannot attach to the active ACT.

The CSI Execution Modes

The CSI application supports the three execution modes that are also supported by ACS.

In batch mode, the standard input file can contain conversation security information for different partner systems. A CSI process can communicate with multiple partner systems (sequentially, one after the other) to handle a batch of CSI application parameters.

The CSI Command Parameters

The CSI command parameters can be divided into these classes:

- The MERVA instance directory parameter
- CSI execution control parameters

- Security information parameters

All ACS parameter classes and parameters apply also to the CSI parameter, however the CSI application provides additional execution control parameter to control a CSI processing trace:

t1 Asks for a normal CSI process trace that does not include the C-Probe sent to a partner system

t2 Asks for an extended CSI process trace that includes the C-Probe sent to a partner system

The CSI writes its processing trace to the file **ekacsi.trace** in the trace subdirectory of the applicable MERVA USS instance directory. The sample CSI trace file name reads **/u/merva1/trc/ekacsi.trace**. Each CSI process overwrites the trace file of a former CSI process (if applicable).

Security Information Parameters

The parameters of this class let you specify conversation security information as CSI command parameters. CSI security information parameters have the format of keyword parameters (a keyword followed by data). A parameter keyword and its data must be specified as two tokens separated by one or more blanks. The security information parameter keywords are:

node identifies the following data as the applicable partner MERVA Link node name. The partner MERVA Link node name identifies the partner system that must use the client user name and password to access the local system.

user identifies the following data as the applicable client user ID (user name) that is defined in the local system.

pswd identifies the following data as the applicable client user password.

For more information about the CSI security information parameters, refer to the section about ACS security information parameters (they are identical) in the *MERVA for ESA Customization Guide*.

Sample CSI Commands

The CSI command:

ekacsi /u/merva1/ h
Sets the USS shell environment variable for the MERVA USS instance directory and displays CSI help information.

ekacsi v c t2
Starts the CSI application in verbose interactive mode, prompts the operator for all security information items, and requests a confirmation before it handles the specified set of security information items. A detailed trace is written to the CSI trace file. This command requires that the USS shell environment variable for the MERVA USS instance directory has been set before the command is executed.

ekacsi node ***pnode1*** **user** ***pn1user*** **pswd** ***pn1pswd***
Starts the CSI application in interactive mode, and handles the complete set of security information items without asking for a confirmation.

The CSI Standard Input File

The CSI application retrieves the conversation security information for a set of partner systems from standard input (**stdin**) if it is started with the control parameter **s**. The format and content of a CSI standard input file is the same as the format and content of an ACS standard input file.

The CSI Batch Mode

Both an OS/390 batch environment and a USS shell environment can host the CSI application in batch mode. The USS shell environment accepts input from the terminal or from an HFS file.

For more information about the CSI batch mode, refer to the description of the ACS batch mode in the *MERVA for ESA Customization Guide*. All ACS batch mode considerations apply to the CSI batch mode as well.

Application Control Daemon for Verification (VCD)

The MERVA Link USS application control daemon used for verification (VCD) is an interactive USS shell application used to verify the correctness of a MERVA Link USS configuration file before it is used to configure an actual MERVA Link USS Gateway. The VCD provides almost the same functionality as the ACD, but can only be used for verification purposes.

A MERVA Link USS configuration file can be verified in the environment of the applicable MERVA USS instance owner (for example, /u/merva1/) or in another USS user's environment (for example, /u/user1). The MERVA USS instance HFS structure applies in both cases.

The verification of a MERVA Link USS configuration file can be divided into several steps:

- Use VCD to scan the specified configuration file, check for formal errors, and write a short summary report to the display terminal.
- Ask VCD to provide a detailed report of all errors and inconsistencies found. The report is written to the specified HFS file or to a file in the specified directory (like the ACD trace).
- Ask VCD to generate an ACT for configuration verification purposes and assume the role of a MERVA Link USS daemon. This ACT can be operated and the VCD can be terminated by the MERVA Link USS Configuration Verification Command Application (VCC). VCC is almost the same as the MERVA Link USS application ACC, but can only be used for verification purposes. For more information about VCC, see "Application Control Command Application for Verification (VCC)" on page 116.

Starting the VCD

The VCD is started by an authorized USS user via a USS shell command or a shell command script.

Examples of Using USS Shell Scripts to Issue the VCD Command

How you use USS shell scripts to issue the VCD command depends on whether you are the MERVA instance owner (that is, if your ID is **merva1**), or if you are another user. If you are the MERVA instance owner, you might use scripts similar to these:

- The following shell script performs the minimum VCD function. It does not write a report, and terminates immediately.

  ```
  /usr/lpp/merva/bin/ekavcd /u/merva1/ cfg /u/merva1/cfg/ekaact.cfg
  ```

- The following shell script checks the specified configuration file, writes a report to the MERVA instance trace directory, and terminates.

```
/usr/lpp/merva/bin/ekavcd      /u/merva1/                  \
                           cfg /u/merva1/cfg/ekaact.cfg     \
                           rep /u/merva1/trc/
```

- The following shell script checks the specified configuration file, writes a report to the MERVA instance trace directory, and assumes the role of a MERVA Link daemon.

```
/usr/lpp/merva/bin/ekavcd      /u/merva1/                  \
                           cfg /u/merva1/cfg/ekaact.cfg     \
                           rep /u/merva1/trc/           k  &
```

  A VCD process that keeps the generated ACT should execute in the USS background (specify & at the end of the command line arguments). Otherwise, the terminal is locked for input until the VCD process ends.

If you are not the MERVA instance owner, you use shell scripts in exactly the same way, except:

- You specify your own home directory (for example **/u/user1/**) instead of that of the MERVA instance owner (**/u/user1/**).
- You must first set up a pseudo MERVA USS instance HFS directory structure. The pseudo MERVA USS instance environment is based on your home directory, which in this example is **/u/user1/**. This HFS structure and the necessary resources can be set up by a USS command sequence similar to the following:

```
cd                    make /u/user1/ the current directory
mkdir ipc             generate IPC subdirectory
cd ipc                make /u/user1/ipc/ the curr directory
touch ekaact.v        generate IPC resource file for VCD/VCC
```

Function of the VCD

When the VCD is called, it checks the VCD parameters, the syntax of the configuration file, and the contents of the configuration file.

Verifying the VCD Command Parameters

Error messages are written to stdout (for example, the terminal) if any of the following are true:

- The MERVA USS instance directory is not specified as the first command parameter.
- The configuration file name parameter is missing.
- The specified configuration file does not exist.
- The specified report directory does not exist.

Verifying the Configuration File Syntax

The formal content of the configuration file is checked by VCD using the same internal MERVA Link USS functions that are used by the ACD when a real ACT must be generated. The most important formal checks are:

- The ACT header parameter group is specified as the first parameter group in the configuration file.
- There is only one ACT header parameter group in the configuration file.
- The parameter group identifiers are valid (ACTH, ACTA, or ACTC).
- The parameter line structure is valid (keyword = value), whereby the parameter value can be optional.

If a syntax error is found in the configuration file and a report is not requested, a single error message is written to sysout. If this happens, start VCD again and request that a report be created. Any error messages will be inserted at the appropriate places in a copy of the configuration file in the report.

If a formal error is found in the configuration file, an ACT is not generated, so VCD does not check the configuration data.

Verifying the Configuration File Data

When the syntax of the configuration file is acceptable, VCD generates an ACT and checks the following:

- The local node name specified in the ACT header
- The processing trace directory path name in the ACT header
- The partner node name specified in each ACT ISC entry
- That each partner node name is specified only in one ACT ISC entry
- Any (SNA APPC or TCP/IP) intersystem connection parameters specified in each ACT ISC entry
- If TCP/IP intersystem connection parameters are specified in an ACT ISC entry, that the messaging port number is greater than 1024

If a configuration data error is found in the ACT and a report is not requested, one or two error messages are written to stdout. If this happens, start VCD again and request a report. Any error or a warning message will be written to the report.

If the VCD parameter **k** is specified, the ACT is kept even if its configuration file contains data errors. The configuration verification command (VCC) can be used to display information in this ACT, and to verify the ACT data error report.

Stopping the VCD

The VCD process terminates immediately if either of the following are true:

- The verification of the VCD command parameters or of the configuration file syntax, described in “Function of the VCD” on page 115, fails.
- The **k** parameter, which indicates that the generated ACT is to be kept active, is not specified.

If the **k** parameter is specified and no errors are found, you must request the termination of the VCD process by doing one of the following:

- Issue the VCC command **trm daemon**. This command is accepted only if issued by the VCD process owner or from a USS root user with uid=0.
- Send an interrupt signal (SIGINT) to the VCD process.

Application Control Command Application for Verification (VCC)

The MERVA Link USS application control command application used for verification (VCC) is almost the same as the ACC, but can only be used for verification purposes. It attaches to the ACT generated by a VCD, and displays the data in that VCD’s ACT as specified by the ACC.

The only major difference between the ACC and VCC is the means by which they attach to the applicable ACT shared memory:

- ACC uses the IPC resource names **ekaact.a** and **ekaact.b** in the /ipc/ subdirectory of the applicable MERVA instance directory.

- VCC uses the IPC resource name **ekaact.v** in the /ipc/ subdirectory of the applicable real or pseudo MERVA instance directory.

VCC can, like ACC, execute in single command mode and in conversation mode. The command syntax for VCC and ACC is identical.

Examples of the MERVA Instance Owner Using VCC

In this example, the MERVA instance owner **merva1** has set the MERVA USS environment variable by adding the statement **export MERVA_DIR=/u/merva1/** to the **.setup** script during installation.

A VCD instance must be active before VCC can be started. In this example, the MERVA instance owner starts the VCD by issuing the command:

```
ekavcd /u/merva1/ cfg /u/merva1/cfg/ekaact.cfg k &
```

The command to list all entries in the ACT generated by the VCD is:

```
ekavcc l
```

The command to enter VCC conversation mode is:

```
ekavcc sc
```

The command to terminate VCD in VCC conversation mode is:

```
trm daemon
```

Examples of Another USS User Using VCC

This example assumes the user **user1** does not have the MERVA USS environment variable set. This user establishes a pseudo MERVA USS instance environment based on his or her home directory **/u/user1/**. This environment is set up by the following USS command sequence:

```
cd                  make /u/user1/ the current directory
mkdir ipc           generate IPC subdirectory
cd ipc              make /u/user1/ipc/ the curr directory
touch ekaact.v      generate IPC resource file for VCD/VCC
```

The user then uses a shell script to call VCC. This shell script is named **vcc** and reads:

```
MERVA_DIR=/u/user1/
/usr/lpp/merva/bin/ekavcc $1 $2 $3
```

A VCD instance must be active before VCC can be started. A sample command to start a VCD instance is:

```
ekavcd /u/user1/ cfg /u/merva1/cfg/ekaact.cfg k &
```

The command to list all entries in the ACT generated by the VCD is:

```
vcc l
```

The command to enter VCC conversation mode is:

```
vcc sc
```

The command to terminate VCD in VCC conversation mode is:

```
trm daemon
```

Local Security Control Application for Verification (VCS)

The local security control application for verification (VCS) provides basically the same service as the local security control application (ACS), but whereas ACS cooperates with an ACD in a production environment, VCS cooperates with a VCD in a test environment.

The differences between VCS and ACS are described in the following. For more information about ACS, refer to the *MERVA for ESA Customization Guide*.

The VCS Program

The command **ekavcs** (or the applicable fully qualified program path name) is used to call the VCS in an OS/390 USS shell environment. The VCS can be executed in an OS/390 TSO USS shell or in a remote login shell at a remote host.

The VCS Execution Environment

The VCS must have access to a MERVA Link USS ACT that is owned by a VCD. This is why the VCD must be active when the VCS is called, and the name of the MERVA USS instance directory must be available to the VCS program.

The fully qualified path name of the MERVA USS instance directory can be provided as the first VCS command parameter. If this parameter is missing when the VCS is called in a USS shell, the USS shell environment variable **MERVA_DIR** must contain the MERVA USS instance directory name. If this environment information is missing or incorrect, the VCS cannot attach to the configuration verification ACT.

The VCS Execution Modes and Command Parameters

The ACS and VCS execution modes and command parameters are the same. ACS executes in a MERVA USS production environment, and attaches to a production ACT (IPC resource names ekaact.a or ekaact.b). VCS executes in a MERVA USS test environment (configuration verification environment), and attaches to a test ACT (IPC resource name is ekaact.v).

The VCS Function

Like the ACS, the VCS generates security files in the **sec** subdirectory of the MERVA USS instance directory, for example, **/u/merva1/sec/**, and updates the security information in the ACT it is attached to. Whereas the ACS is attached to an ACT that represents a MERVA Link USS production environment, the VCS is attached to an ACT that represents a MERVA Link USS test environment. The security files, however, apply to both the production and the test environments if the MERVA USS instance directory is the same for both the production and test environments.

If the MERVA USS instance directory is the same for ACS and VCS, the conversation security information specified via VCS does not become immediately active for the production environment. It can be activated by restarting the ACD in the production environment.

Chapter 10. MERVA Link USS in the OS/390 USS Environment

The OS/390 UNIX System Services (USS) Environment is the primary environment for MERVA Link USS. This means that the MERVA Link executables (shell scripts, programs, and DLLs) reside in the OS/390 Hierarchical File System (HFS), and are executed in a child process of a USS shell (for example, /bin/sh) or a daemon (for example, /usr/sbin/inetd).

Parameters can be passed to the main program of a process implicitly and explicitly. The explicit parameters can be retrieved from the program arguments list. The implicit parameters can be retrieved from the list of environment parameters. MERVA Link USS programs provide for specifying some mandatory parameters alternatively as an explicit or as an implicit parameter. You may choose the most convenient method in these cases.

The following sections describe various aspects of MERVA Link USS in the OS/390 USS environment.

Standard MERVA Link USS Program Call Environment

The standard MERVA Link USS program call environment has various aspects.

MERVA USS Environment Variables

Two OS/390 USS environment variables are defined for MERVA USS when MERVA USS is installed and when a MERVA USS instance is generated.

DSLPP_DIR
: Defines the MERVA USS installation directory. The sample MERVA USS installation directory is **/usr/lpp/merva**. MERVA USS can be installed more than once in an OS/390 system. You may wish, for example, to use one installation for production, and another installation to test MERVA USS program PTFs, or a new MERVA USS release.

MERVA_DIR
: Defines the MERVA USS instance directory. The sample MERVA USS instance directory is **/u/merva1**. The MERVA USS instances generated in an OS/390 system can be based on the same or on different MERVA USS installations.

MERVA Link USS programs may refer to these environment variables in order to establish the processing environment for a MERVA Link USS process. This is why these variables should be defined in the USS shell of any user who works with MERVA Link USS.

PATH and LIBPATH Environment Variables

The OS/390 USS shell uses the sequence of directories specified in the PATH environment variable to locate the executable of a command or of a called program. The LIBPATH environment variable is used by the OS/390 USS dynamic loader to locate the DLL that contains a called function.

In the standard MERVA Link USS program call environment, the PATH and LIBPATH environment variables need not contain MERVA USS installation subdirectories if you follow these rules:

Specify the full path of the executable

The full path of the executable should always be specified when a MERVA Link USS program is called. The USS shell does not refer to the PATH variable in this case.

During the MERVA USS instance generation a number of shell scripts and command aliases are made available that provide for minimum typing when MERVA Link programs are called in a USS shell. The full path names are contained in the shell scripts or in the command alias definitions, and you need not type long program path names.

Use the standard MERVA USS installation directory

The full path of an executable should be part of a standard MERVA USS installation directory tree. The MERVA USS installation directory can be any HFS directory. To be a standard MERVA USS installation directory, it must have a **bin** and a **lib** subdirectory containing the MERVA Link USS programs and DLLs, respectively.

If programs are called from a standard **bin** subdirectory, the corresponding **lib** subdirectory can be easily evaluated, and the LIBPATH environment variable can be set by the MERVA Link USS program as appropriate.

If a MERVA Link USS program is not called from a standard **bin** subdirectory, it may use other information (for example, the DSLPP_DIR variable) to set the LIBPATH. If the information needed to set the LIBPATH is missing, successful execution depends on the setting of the LIBPATH before the MERVA Link USS program was called.

MERVA Link USS Program Call Environments

MERVA Link USS programs execute in four different environments:

- OS/390 USS Shell
- OS/390 USS InetD Subserver Process
- OS/390 BPXBATCH Process (APPC/MVS)
- OS/390 MVS Region

Most of the MERVA Link USS programs can execute in more than one of these environments. The MERVA Link USS programs and the applicable environments are discussed in the following sections.

The OS/390 MVS environment is, however, not covered because the following considerations don't apply for the OS/390 MVS environment. The environment of a program executing in an OS/390 MVS region is completely established by the MVS JCL. For more information about MERVA Link USS programs executing in the OS/390 MVS environment, refer to "Chapter 11. MERVA Link USS in the OS/390 MVS Environment" on page 123.

MERVA Link USS Programs Called in a USS Shell

The following MERVA Link USS programs are called from within a USS shell:

ekaacd	Application control daemon
ekaacc	Application control command application
ekaacs	Local security control application
ekacsi	Partner security control application
ekavcd	Application control daemon used for verification
ekavcc	Application control command application used for verification

ekavcs	Security control application used for verification

You can start one of these programs from within a USS shell by calling a shell script. Before running the program, the shell sets the value of the program path environment variable _ to the path and name of the program being called, for example **/usr/lpp/merva/bin/ekavcd**.

The LIBPATH is set as follows:

- If the program being called resides in a subdirectory with the name **bin**, the program sets the LIBPATH to a subdirectory on the same level as its **bin** subdirectory, but with the name **lib**, for example **/usr/lpp/merva/lib**.
- If it does not reside in a subdirectory with the name **bin**, the program checks whether the environment variable DSLPP_DIR is set. If it is, the program sets the LIBPATH to the **lib** subdirectory of the directory specified DSLPP_DIR.
- If it does not reside in a subdirectory with the name **bin** and the environment variable DSLPP_DIR is not set, you must set the LIBPATH to include the directory containing the program DLLs before calling the program.

MERVA Link USS Inbound SNA APPC TP (ekatpi)

The MERVA Link USS inbound SNA APPC TP **ekatpi** can be started either directly or indirectly (that is, by means of a shell script) in an OS/390 USS BPXBATCH environment. If started directly, environment variables can be passed to ekatpi in the BPXBATCH environment file (DD name is STDENV). If started indirectly (by means of a shell script), environment variables can be set in the shell script before calling the program.

When the BPXBATCH utility or the shell script calls ekatpi, it must include the path to the directory in which the program resides, for example **/usr/lpp/merva/bin/ekatpi**. The program receives this string as its first argument (arg0).

The LIBPATH is set as follows:

- If ekatpi resides in a subdirectory with the name **bin**, ekatpi sets the LIBPATH to a subdirectory on the same level as its **bin** subdirectory, but with the name **lib**, for example **/usr/lpp/merva/lib**.
- If ekatpi does not reside in a subdirectory with the name **bin**, ekatpi checks whether the environment variable DSLPP_DIR is set. If it is, ekatpi sets the LIBPATH to the **lib** subdirectory of the directory specified DSLPP_DIR.
- If ekatpi does not reside in a subdirectory with the name **bin** and the environment variable DSLPP_DIR is not set, you must set the LIBPATH to include the directory containing the ekatpi DLLs before calling ekatpi.

MERVA Link USS Inbound TCP/IP TP (ekatci)

The MERVA Link USS Inbound TCP/IP TP **ekatci** is started directly or indirectly (via a shell script) by the OS/390 USS Internet Daemon (InetD) as an InetD subserver. InetD sets the PATH environment variable to the path of the subserver executable as specified in the subserver's entry in the InetD configuration file (**/etc/inetd.conf**). This path is either the directory path to ekatci (direct call) or the directory path to the shell program (**/bin/sh**).

Before running the program, the shell sets the value of the program path environment variable _ to the path and name of the program being called, for example **/usr/lpp/merva/bin/ekatci**.

To set the LIBPATH, ekatci checks the environment variables _ and **PATH**:

- If the program path (environment variable _) ends with **/bin/ekatci**, the LIBPATH is set to a subdirectory on the same level as this **bin** subdirectory, but with the name **lib**.
- If the program path does not end with **/bin/ekatci**, and if the value of the PATH environment variable ends with **/bin** and is longer than four characters, the LIBPATH is set to a subdirectory on the same level as this **bin** subdirectory, but with the name **lib**.

Chapter 11. MERVA Link USS in the OS/390 MVS Environment

The OS/390 UNIX System Services (USS) environment is the primary environment for MERVA Link USS. This means that the MERVA Link executables (shell scripts, programs, and DLLs) reside in the OS/390 Hierarchical File System (HFS) and are, in most cases, executed via an USS shell command.

The OS/390 MVS environment is the alternate environment for MERVA Link USS. This means that the MERVA Link executable program objects (programs and DLLs) reside in an OS/390 PDSE, and are, in most cases, executed via an MVS batch job.

MERVA Link USS processes that are started from a PDSE member finally also execute in an OS/390 USS environment. An MVS process is dubbed to a USS process automatically by OS/390 if it requests USS services. This is what all MERVA Link USS processes do to access the ACT SHM and to process HFS files, for example.

The following sections describe various aspects of MERVA Link USS in the OS/390 MVS environment.

Allocate MERVA Link USS MVS Data Sets

Two MVS data sets are used to install MERVA Link USS executables in the OS/390 MVS environment:

- The OS/390 Binder Side Definition data set **SYSDEFSD** contains IMPORT control statements for all exported functions of all MERVA Link USS DLLs.
- The OS/390 Program Object Library contains all MERVA Link USS executables (programs and DLLs).

Side Definitions Data Set

The side definitions data set is a standard PDS with fixed-length record members. The OS/390 binder creates a side definition file (PDS member) and writes IMPORT control statements to that file when it creates a DLL. The name of the side definition file is the same as the name of the DLL. When the MERVA Link USS DLLs have been created, this data set contains one member for each DLL.

The side definitions data set must have the attributes RECFM=F or RECFM=FB, LRECL=80. The sample MERVA Link USS side definitions PDS has the name **hlq.SDSLIMP**, where **hlq** is the high-level qualifier of your MERVA ESA installation. The record format is FB with a record length of 80 and a block size of 400. The MERVA Link USS IMPORT files occupy about 32 blocks in this data set.

Program Objects Data Set

The program objects data set is an OS/390 PDSE. The OS/390 binder creates a program object (PDSE member) and writes it to that data set when it creates a DLL or a program.

The program objects data set must have the attribute RECFM=U. The sample MERVA Link USS program library PDSE is named ***hlq*.SDSLLIB**, where ***hlq*** is the high-level qualifier of your MERVA ESA installation. The record format is U with a record length of 0 and a block size of 4096. The MERVA Link USS program objects occupy about 512 blocks in this data set.

A PDSE can be allocated in a TSO/E session using option M (enhanced data set allocation) in the ISPF Data Set Utility (PDF 3.2).

Copy DLLs from HFS Library to PDSE

In the following samples, the MERVA Link USS dynamic link libraries (DLLs) are assumed to be contained in the HFS directory **/usr/lpp/merva/lib/**. All DLLs must be copied to a program object library if MERVA Link USS is to execute in the OS/390 MVS environment. The OS/390 binder (program IEWBLINK) is used for that pupose.

A DLL may use functions that are exported by other DLLs. The IMPORT files of these other DLLs must be included when a DLL is copied from the HFS to the PDSE. An INCLUDE file for the copied DLL is created by the binder during this process. The INCLUDE file contains INCLUDE statements for all functions exported by the copied DLL.

When the set of MERVA Link USS DLLs is copied the first time, INCLUDE files are not available from the side definitions data set. This is why the DLLs that export functions to other DLLs must be copied first.

Copy DLL Procedure

The following sample procedure can be used to copy DLLs from the HFS to a PDSE:

```
//LINKDLL  PROC DLL=EKADLL
//LINK     EXEC PGM=IEWBLINK,REGION=4M,
//         PARM='LIST,LET,RENT,REUS,DYNAM(DLL),CASE=MIXED'
//SYSPRINT DD SYSOUT=*
//INLIB    DD PATH='/usr/lpp/merva/lib/'
//INIMP    DD DSN=hlq.SDSLIMP,DISP=SHR
//SYSDEFSD DD DSN=hlq.SDSLIMP(&DLL),DISP=SHR
//SYSLMOD  DD DSN=hlq.SDSLLIB,DISP=SHR
//         PEND
```

The binder option **DYNAM(DLL)** controls DLL processing and must be specified.

The data sets specified in the DD statements do the following:

- INLIB specifies the HFS directory that contains the DLL to be copied.
- INIMP specifies the PDS that contains INCLUDE files for functions exported by other DLLs.
- SYSDEFSD specifies the name of the INCLUDE file that is generated by the binder. This file contains INCLUDE statements for all functions exported by the subject DLL.
- SYSLMOD specifies the name of the target program object library.

This procedure is used in the copy jobs described in the next section.

Copy DLL Job Sequence

DLLs that call functions provided by other DLLs must be copied after copying the latter DLLs. A sample sequence for copying the MERVA Link USS DLLs is:

1. ekacex

2. ekaeex
3. ekacsc (ekacex, ekaeex)
4. ekadsc (ekacsc)
5. ekatco (ekacsc)
6. ekatpo (ekacsc)
7. ekaasi (ekacsc)
8. ekap2i (ekacsc, ekaasi, ekap1o)
9. ekap1i (ekacsc, ekatco, ekatpo, ekap2i)
10. ekap1o (ekacsc, ekatco, ekatpo)
11. ekap2o (ekacsc, ekap1o)

The DLLs that export functions for the subject DLLs are shown in parentheses after the HFS names of the subject DLLs.

Sample Copy DLL Job Steps

The following job step can be used to copy DLL **ekacsc** from the HFS to a PDSE as program object **EKACSC**:

```
//EKACEX   EXEC LINKDLL,DLL=EKACEX
//SYSLIN   DD *
   INCLUDE INLIB(ekacex)
   NAME    EKACEX(R)
/*
```

This job copies the MERVA Link USS Common Error Explanation DLL to the MERVA Link USS program object library, and writes the IMPORT statements to the EKACEX side definition file (member EKACEX of SYSDEFSD).

The following sample job step can be used to copy DLL **ekacex** from the HFS to a PDSE as program object **EKACEX**:

```
//EKAP1I   EXEC LINKDLL,DLL=EKAP1I
//SYSLIN   DD *
   INCLUDE INLIB(ekap1i)
   INCLUDE INIMP(EKACSC,EKATCO,EKATPO,EKAP2I)
   NAME    EKAP1I(R)
/*
```

This job copies the MERVA Link USS Inbound P1 processor to the MERVA Link USS program object library, and writes the IMPORT statements to the EKAP1I side definition file (member EKAP1I of SYSDEFSD).

The copy job steps for the other MERVA Link USS DLLs can be created accordingly.

Copy Programs from HFS Library to PDSE

In the following samples, the MERVA Link USS programs are assumed to be contained in the HFS directory **/usr/lpp/merva/bin/** when MERVA USS has been installed. All programs that must be executed in the OS/390 MVS environment must be copied to a program object library. The OS/390 Binder (program IEWBLINK) is used for that pupose.

The MERVA Link USS programs use functions that are exported by DLLs. The IMPORT files of these DLLs must be included when a program is copied from the HFS to the PDSE.

The set of MERVA Link USS DLLs must be copied as described in “Copy DLLs from HFS Library to PDSE” on page 124 before the MERVA Link USS programs can be copied. INCLUDE files required for copying programs may not be available from the side definitions data set before all DLLs are copied.

Copy Program Procedure

The following sample procedure can be used to copy MERVA Link USS programs from the HFS to a PDSE:

```
//LINKPGM  PROC
//LINK     EXEC PGM=IEWBLINK,REGION=4M,
//         PARM='LIST,LET,RENT,REUS,DYNAM(DLL),CASE=MIXED'
//SYSPRINT DD SYSOUT=*
//INLIB    DD PATH='/usr/lpp/merva/bin/'
//INIMP    DD DSN=hlq.SDSLIMP,DISP=SHR
//SYSLMOD  DD DSN=hlq.SDSLLIB,DISP=SHR
//         PEND
```

The binder option **DYNAM(DLL)** controls DLL processing and must be specified.

The data sets specified in the DD statements do the following:

- INLIB specifies the HFS directory that contains the program to be copied.
- INIMP specifies the PDS that contains INCLUDE files for functions exported by MERVA Link USS DLLs.
- SYSLMOD specifies the name of the target program object library.

This procedure is used in the copy jobs described in the next section.

Copy Program Job Sequence

The MERVA Link USS programs can be copied in any sequence after copying the DLLs that export functions to the subject program. The MERVA Link USS programs and the directly called DLLs are:

- EKAACC (EKACSC, EKACEX, EKAEEX, EKADSC, EKAP1O)
- EKAACS (EKACSC)
- EKAACD (EKACSC, EKADSC)
- EKAASO (EKACSC, EKAP2O)
- EKACSI (EKACSC, EKAP1O)
- EKATCI (EKACSC, EKAP1I)
- EKATPI (EKACSC, EKAP1I)
- EKAVCC (alias of EKAACC)
- EKAVCD (EKACSC, EKADSC)
- EKAVCS (alias of EKAACS)

The DLL program objects that export functions for the subject programs are shown in parentheses after the PDSE names of the subject programs. The programs EKAVCC and EKAVCS are aliases of EKAACC and EKAACS, respectively.

All MERVA Link USS programs except the InetD subserver program EKATCI can be executed in the OS/390 MVS environment. It may make no sense, therefore, to copy program EKATCI from the HFS to a PDSE.

Sample Copy Program Job Steps

The following sample job step can be used to copy program **ekaacc** from the HFS to a PDSE as program object **EKAACC**:

```
//EKAACC   EXEC LINKPGM
//SYSLIN   DD *
   INCLUDE INLIB(ekaacc)
   INCLUDE INIMP(EKACSC,EKACEX,EKAEEX,EKADSC,EKAP10)
   ALIAS   EKAVCC
   NAME    EKAACC(R)
/*
```

This job copies the MERVA Link USS ACC application program to the MERVA Link USS program object library, and defines **EKAVCC** as an alias name for that program. This alias name corresponds to the symbolic link defined in the HFS for the VCC application program.

The following sample job step can be used to copy program **ekaacd** from the HFS to a PDSE as program object **EKAACD**:

```
//EKAACD   EXEC LINKPGM
//SYSLIN   DD *
   INCLUDE INLIB(ekaacd)
   INCLUDE INIMP(EKACSC,EKADSC)
   NAME    EKAACD(R)
/*
```

This job copies the MERVA Link USS daemon program to the MERVA Link USS program object library. You can create other, similar copy job steps for the other MERVA Link USS programs.

Execute Programs in the OS/390 MVS Environment

This section contains sample MVS jobs that can be used to execute MERVA Link USS programs in the OS/390 MVS environment. OS/390 MVS environment means that the MERVA Link program executes as part of an MVS batch job, or as an APPC/MVS TP started in an APPC/MVS initiator.

Application Control Daemon (ACD)

An MVS batch job can be used to start the ACD. The duration of this job may be indefinite. The CPU requirements are marginal.

An ACD job can be canceled by the standard MVS means, however to terminate an ACD job in an orderly way, do one of the following:

- Send a SIGINT or SIGTERM to the ACD process in the USS environment. The ACD process identifier that must be specified in the kill command is recorded in the ACT header.
- Issue the ACC command ’trm daemon’. This is the most appropriate means to stop an ACD job.

An ACD can be refreshed with an updated configuration in the USS environment by starting another instance of the daemon process. However, an ACD refresh cannot be initiated in the MVS environment by submitting the same ACD job again. The same ACD job will be queued by JES until the active ACD job terminates. It may be possible to refresh an ACD by submitting an ACD job with a job name that is different from the name of the active ACD job.

The following is an example of an ACD job:

```
//xxxxxxxx JOB xxxxxxxxxxxxxxxxxxx
//EKAACD   EXEC PGM=EKAACD,REGION=4M,
//         PARM='//u/merva1/ cfg /u/merva1/cfg/ekaacd.cfg
//               trc /u/merva1/trc/'
//STEPLIB  DD DSN=hlq.SDSLLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//
```

The PARM parameter of this job specifies the MERVA instance directory **/u/merva1/**, the ACD configuration file **/u/merva1/cfg/ekaacd.cfg**, and the ACD trace directory **/u/merva1/trc/**.

SYSPRINT represents the standard output (stdout) of the ACD process. The ACD writes error information to stdout if an error occurs before it can open the trace file.

Application Control Command Application (ACC)

The ACC is primarily an interactive application, but it also accepts input commands from a batch file. Batch input mode is the only reasonable way in which to execute ACC in an MVS environment.

The following is an example of an ACC job:.

```
//xxxxxxxx JOB xxxxxxxxxxxxxxxxxxx
//EKAACC   EXEC PGM=EKAACC,REGION=4M,PARM='/si /u/merva1/'
//STEPLIB  DD DSN=hlq.SDSLLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
dph
la
lc
dpa *
dpc *
/*
//
```

The PARM parameter of this job specifies that the ACC must be started in batch input mode (**si**), and specifies the name of the MERVA instance directory (**/u/merva1/**).

SYSPRINT represents the standard output (stdout) of the ACC process. ACC writes command output to stdout. SYSIN represents standard input (stdin). ACC reads commands from stdin.

Local Conversation Security Application (ACS)

The ACS is primarily an interactive application, but it also accepts input data from **stdin**. The ACS standard input mode is the only reasonable way in which to execute ACS in an MVS environment.

The following is an example of an ACS job:

```
//xxxxxxxx JOB xxxxxxxxxxxxxxxxxxx
//EKAACS   EXEC PGM=EKAACS,REGION=4M,
//         PARM='//u/merva1/ s v'
//STEPLIB  DD DSN=hlq.SDSLLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
#------------------------------------------
# MERVA Link USS Conversation Security Info
#------------------------------------------
# Partner Node   Client User Name   Password
```

```
#------------------------------------------
NODE2           user2             passwd2
NODE3           user3             passwd3
/*
//
```

The PARM parameter of this job specifies that the ACS must be started in verbose mode (**v**), and that it must read data from stdin (**s**). The name of the MERVA instance directory is specified as **/u/merva1/**.

SYSPRINT represents the standard output (stdout) of the ACS process. ACS writes operator messages to stdout. SYSIN represents standard input (stdin). ACS reads conversation security data from stdin.

Partner Conversation Security Application (CSI)

The CSI is primarily an interactive application. It supports, however, also input data from **stdin**. The CSI standard input mode is the only reasonable way in which to execute CSI in an MVS environment.

The following is an example of a CSI job:.

```
//xxxxxxxx JOB xxxxxxxxxxxxxxxxxxx
//EKACSI   EXEC PGM=EKACSI,REGION=4M,
//         PARM='//u/merva1/ s t2'
//STEPLIB  DD DSN=hlq.SDSLLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
#------------------------------------------------
# MERVA Link USS Conversation Security Info
#------------------------------------------------
# Partner Node   Local Client User Name   Password
#------------------------------------------------
NODE2           user2                   passwd2
NODE3           user3                   passwd3
/*
//
```

The PARM parameter of this job specifies that the CSI application must read data from stdin (**s**), and that it must write a detailed trace (**t2**) to the HFS file **/u/merva1/trc/ekacsi.trace**. The name of the MERVA instance directory is specified as **/u/merva1/**.

SYSPRINT represents the standard output (stdout) of the CSI process. CSI writes operator messages to stdout. SYSIN represents standard input (stdin). CSI reads conversation security data from stdin.

Application Control Daemon for Verification (VCD)

An MVS batch job can be used to start the VCD. This batch job is similar to the job used to start the ACD; the only difference is that the program name is EKAVCD instead of EKAACD.

A VCD can verify the correctness of a MERVA Link USS configuration file and generate an ACT for test purposes without affecting an active MERVA Link instance. The VCD and its test ACT do not support the transfer of messages. All other ACD and ACC functions apply to VCD and VCC also. For more information refer to “Application Control Daemon (ACD)” on page 127.

The following is an example of a VCD job:

```
//xxxxxxxx JOB xxxxxxxxxxxxxxxxxxx
//EKAVCD   EXEC PGM=EKAVCD,REGION=4M,
//         PARM='//u/merva1/ cfg /u/merva1/cfg/ekaacd.cfg
//              trc /u/merva1/trc/'
//STEPLIB  DD DSN=hlq.SDSLLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//
```

Application Control Command Application for Verification (VCC)

The VCC attaches to the test ACT generated by the VCD. Like the ACC, it is primarily an interactive application, but it also accepts input commands from a batch file. Batch input mode is the only reasonable way in which to execute the VCC in an MVS environment.

The following is an example of a VCC job:

```
//xxxxxxxx JOB xxxxxxxxxxxxxxxxxxx
//EKAVCC   EXEC PGM=EKAVCC,REGION=4M,PARM='/si /u/merva1/'
//STEPLIB  DD DSN=hlq.SDSLLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
dph
la
lc
dpa *
dpc *
trm daemon
/*
//
```

The PARM parameter of this job specifies that the VCC must be started in batch input mode (**si**), and that the name of the MERVA instance directory is **/u/merva1/**.

SYSPRINT represents the standard output (stdout) of the VCC process. VCC writes command output to stdout. SYSIN represents standard input (stdin). VCC reads commands from stdin.

The last VCC command in this sample requests the termination of the VCD.

Local Conversation Security Application for Verification (VCS)

The VCS attaches to ACT generated and owned by the VCD. It can generate security files that are usable by an ACD and a VCD, and modifies conversation security information in its ACT (the ACT owned by the VCD). Conversation security information in the ACT of an ACD cannot be modified by the VCS.

All other functions of the ACS apply also to VCS. For more information refer to "Local Conversation Security Application (ACS)" on page 128.

The following is an example of a VCS job:

```
//xxxxxxxx JOB xxxxxxxxxxxxxxxxxxx
//EKAVCS   EXEC PGM=EKAVCS,REGION=4M,
//         PARM='//u/merva1/ s v'
//STEPLIB  DD DSN=hlq.SDSLLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
#-----------------------------------------
# MERVA Link USS Conversation Security Info
#-----------------------------------------
# Partner Node   Client User Name   Password
```

```
#------------------------------------------
NODE2           user2             passwd2
NODE3           user3             passwd3
/*
//
```

Inbound SNA APPC TP (TPI)

When APPC/MVS schedules a TP from a PDSE, the started transaction does not execute in an USS environment initially. This is why the TP profile and the TP itself must care for the mandatory USS environment variables.

The following is a sample of JCL in a TP profile for MERVA Link USS (EKARP1) that executes EKATPI as a PDSE member:

```
 TP name: EKARP1
 Level  : SYSTEM          ID . . . :
******************************* Top of Data ************************
//xxxxxxx  JOB  xxxxxxxxxxxxxxxxxxxxxxxxx
//EKAR1    EXEC PGM=EKATPI,
//              PARM='//u/merva1/ TZ=CET-1 '
//STEPLIB  DD   DSN=hlq.SDSLLIB,DISP=SHR
//SYSPRINT DD   DSN=stdout,DISP=SHR
****************************** Bottom of Data **********************
```

This TP profile defines the MERVA Link USS receiving TP that is associated with MERVA USS instance 1. It can receive messages from all kinds of MERVA Link partner systems.

The format and content of the JOB statements depends on requirements of the OS/390 installation.

The //EKAR1 EXEC statement starts the job step and identifies the program to be executed (EKATPI). Parameters must be passed to EKATPI as follows:

- The first forward slash (/) must be specified to identify the end of the runtime options (there are none in this example).
- /u/merva1/ is the first program parameter (arg1). It specifies the name of the OS/390 USS directory that contains the MERVA Link USS Inter-Process Communication (IPC) resources. IPC resources are used by MERVA Link USS programs to attach to central MERVA Link USS resources, for example, the MERVA Link USS customization data in the MERVA Link USS ACT.
- TZ=CET-1 is the second program parameter (arg2). It specifies the local time zone in the format of the applicable OS/390 USS environment variable. The time zone value is specified as a number of hours west of GMT (London). The New York time zone is identified by TZ=CET+5. The Frankfurt time zone is identified by TZ=CET-1.

  If the time zone parameter is missing or incorrect, the timestamps in MERVA Link USS traces may be incorrect.

The SYSLIB DD statement identifies the location of the executable MERVA Link inbound TP. Program EKATPI resides in an OS/390 program library (PDSE).

The SYSPRINT DD statement defines the standard output file (stdout) to be used by EKATPI. It is an optional statement. Under normal circumstances, no data is written to stdout.

Part 4. MERVA Link Problem Determination Aids

Chapter 12. MERVA Link ESA Conversation Traces

The communication between the local and the remote message transfer processors is performed via PDUs (Protocol Data Units) and APPC indicators. The PDUs are transmitted in segments as specified by the MERVA Link P1 protocol. APPC indicators are transmitted as requested by APPC commands and parameters in these commands following the SNA LU 6.2 protocol.

MERVA Link provides facilities to trace the data exchanged between two APPC processes. These facilities are called the MERVA Link external conversation trace and the MERVA Link internal conversation trace. In this context, external and internal mean the target of the trace. The external trace is written to MERVA Link external resource, a sequential data set. The internal trace is written to a MERVA Link storage area in wrap-around mode.

Internal and external conversation traces are not supported by the MERVA Link TP Mirror EKATM10 that is used in the MERVA Link Back-to-Back environment.

Conversation Trace Control Information

A conversation trace record contains, among other information, the command type and the compressed conversation indicators. These two conversation control information items are described in this section and referred to in the following sections.

The Conversation Command Type

The conversation command type that identifies a command issued by a MERVA Link message transfer program is the most important information in a conversation trace record. It can be an APPC command or a CICS system administration command:

- For an APPC command, it consists of a letter that identifies an APPC command and, if applicable, a specific command parameter.
- For a CICS system administration command, it consists of a digit that identifies the CICS command and, if applicable, the place where the command is issued.

The conversation command identifiers (CIDs), the corresponding commands, and information about each command are shown in Table 3 on page 136.

Table 3. Conversation Trace Command Identifiers

CID	Command	Description
1	Inquire Connection	A CICS APPC MTP checks the status of the connection to the applicable partner system. Follow-on actions can be based on the result of this check.
2	Acquire Connection (begin)	A CICS APPC MTP tries to acquire a connection to the applicable partner system at the begin of a sending task if a connection is not available and if this function is requested by the MTP parameter CONNECT=(ACQ,QUEUE).
8	Acquire Connection (end)	A CICS APPC MTP tries to acquire a connection to the applicable partner system at the end of a sending task if the request to allocate a session failed and if this function is requested by the MTP parameter CONNECT=(ACQ).
9	Release Connection	A CICS APPC MTP releases the connection to the partner system at the end of a sending task if this function is requested by the MTP parameter CONNECT=(REL).
A	Allocate Session	In the APPC/MVS environment, an ALLOCATE command includes the request to attach the remote process.
P	Connect Process	In the CICS environment, the request to attach the remote process is specified in a separate command.
S	Send Data Segment	Send application data (a PDU segment), or error data (a report data element) without a request for confirmation. The application or error data identifier is also shown in the trace record.
T	Send Trailer	Send a PDU trailer without a request for confirmation. The identifier of the PDU trailer is also shown in the trace record.
V	Send Trailer, Confirm	Send a PDU trailer with a request for confirmation, or send a request for confirmation without data. The data identifier in the trace record is 0000 if no data (PDU trailer) is sent.
I	Send Invite, Invite to Send	An inbound TP returns error data to the outbound TP and returns the permission to send to the partner process. For a Send Invite call the identifier of the error data is also shown in the trace record.
R	Receive Data Segment	Conversation control indicators received with this command are shown in the trace record. If application data has been received, the application data identifier is also shown in the trace record.
X	Receive Error	An outbound TP receives error data from the inbound TP. The identifier of the error data is also shown in the trace record.
C	Confirmed	An inbound TP confirms a conversation as response to a request for confirmation received from the partner process.
E	Indicate Error	An inbound TP indicates to the outbound TP that an error condition has been detected. The inbound TP will try to send details of the error condition in an error report to the outbound TP.
F	Free Session	The conversation is terminated and the session to the partner system is freed (deallocated).

Compressed Conversation Indicators

Conversation indicators are used in an APPC conversation to control the state of the processes involved in this conversation. The indicators tell these processes in what state they are, this means, what they can do or must do.

Conversation indicators are found by an outbound TP or by an inbound TP at various places in the CICS and the APPC/MVS APPC interfaces. This is why

MERVA Link collects the most interesting conversation indicators from different places and puts them into a single byte. This byte contains the MERVA Link compressed conversation indicators that are independent of the APPC interfaces provided by CICS or APPC/MVS.

Five conversation indicators are collected by MERVA Link. Each of these indicators is represented by one bit in the MERVA Link Compressed Conversation Indicators byte. The bits of the compressed conversation indicator byte, their meaning and comments for each indicator are shown in Table 4.

Table 4. Compressed Conversation Indicators

CCI Bit	Indicator	Comments
00010000	Receive required	This indicator tells an APPC program that it must issue a RECEIVE command.
00001000	Confirmation required	This indicator tells an APPC program that it must confirm the conversation or raise the error flag.
00000100	Error indicated	An error has been indicated by the partner process. The local process must receive error information.
00000010	No data received	No application data has been received. It is only conversation control information that has been received.
00000001	Free session required	The APPC program must free (deallocate) the session to the partner system.

The External Conversation Trace

The MERVA Link external conversation trace is written by a sending and by a receiving message transfer program to a sequential data set. It is supported in the MERVA Link CICS environment only.

Using the External Conversation Trace

A trace entry is written as soon as control is returned to the MTP after issuing the corresponding APPC command. The conversation trace data set is defined as a CICS output extrapartition transient data queue. The name of this queue is specified in the partner table header (applicable to all receiving MTPs and default for all sending MTPs) or in an EKAPT TYPE=MTP entry (applicable for this message transfer process as sending MTP only).

The MERVA Link conversation trace can be disabled or reset dynamically by closing the data set using the CICS CEMT transaction. The conversation trace is enabled again by opening that data set again. Disable applies immediately for all MTPs that use this data set, including the currently active MTPs. Enable applies for all MTPs that use this data set, excluding the currently active MTPs.

An entry (record) in the MERVA Link conversation trace consists of two parts. First, a fixed-length control information part, and second, a variable-length application data part. The second part is missing in a conversation trace record if there is no application data (for example, with an Allocate Session, Connect Process, or Issue Confirmation). It is also missing if a weak conversation trace was requested via the corresponding parameter in the partner table.

Application data is traced with a maximum length of 2048 bytes. PDU parts sent to or received from the partner process that are greater than 2048 bytes are truncated to that length in the MERVA Link conversation trace. The length of the most interesting parts of a PDU (envelope, heading) is smaller than 2048 bytes. Therefore, this restriction applies only to body parts exceeding the size of 2048 bytes.

In the MVS environment, the MERVA Link conversation trace data set can be printed in dump format (showing both, the hexadecimal and the character representation of the records) using the VSAM utility IDCAMS, although the Conversation Trace data set is not a VSAM data set. In the VSE environment, DITTO can be used instead of IDCAMS.

Interpreting the Control Information Part

The format of the fixed-length control information part of an external conversation trace record, this is, the displacement of its fields in the record, the field lengths, and a description of each field is shown in Table 5 on page 139.

| For CICS commands related to APPC, the first of the two additional control bytes
| (Displ. 50) contains the CICS conversation state (CVDA content).

| As the default, the first 4 bytes of the Additional Control Information field show
| the CICS EIBRCODE, and the second 4 bytes show the CICS EIBERRCD.

| The Additional Control Information field contains the APPC LU name of the
| partner system in the trace of a CICS Inquire Connection command.

| The Additional Control Information field contains the local identification of the
| partner system in an ALLOCATE command trace. It contains the partner process
| identifier in a CICS CONNECT PROCESS command trace.

The Additional Control Information field contains the event return code (2 characters) and the event diagnostic code (6 characters) when an error report (data element ID 1500) is sent.

Table 5. External Conversation Trace Record Control Data

Displ.	Length	Field Description
00	2	Conversation trace record length
02	2	Conversation trace record identifier X'0010'
04	2	Conversation trace control data length
06	2	Conversation trace control data ID X'8110'
08	8	Relative time in the format SSS.FFF- (seconds and milliseconds)
16	8	Internal message transfer process name (send), or partner system identifier (APPC receive)
24	8	Task number and CICS connection status (CVDA value returned by Inquire Connection), or CICS command response value (Acquire Connection), or conversation identifier (APPC related command).
32	1	Command identifier (see Table 3 on page 136)
33	1	Compressed conversation indicators (see Table 4 on page 137)
34	4	Application data identifier in characters, the identifier of the first level-1 data element of the PDU segment. 0000 if no data is associated with this command.
38	2	Byte 3 of EIBRESP and byte 0 of EIBRCODE (CICS), or ATB_RETURN_CODE as a halfword (APPC/IMS)
40	8	Command duration in the format -SS.FFF- (seconds and milliseconds)
48	2	Compressed conversation indicators in characters (see Table 4 on page 137)
50	2	Additional control bytes containing specific information in specific situations
52	4	Total body text length when an AMPDU heading is sent or received
56	8	Additional control information as specified by specific commands in specific situations.

External Conversation Trace Samples

The samples of full and weak external conversation traces shown in the following sections have been obtained from the transfer of messages between the sample MERVA Link systems CICS 1 and CICS 2. Two application messages have been transferred from CICS 1 to CICS 2 in one conversation, and two acknowledgment messages have been returned by CICS 2 in another conversation. The MIP window sizes in CICS 1 and CICS 2 are 010 and 001, respectively.

Full External Conversation Trace Samples

The full conversation trace samples have been generated by MERVA Link transactions in CICS 1.

Example 1: Send Two Application Messages

The following is an example of a full external conversation trace written by a sending MTP during the transfer of two application messages:

```
RECORD SEQUENCE NUMBER - 1
000000  00400010 003C8110 F3F2F84B F6F0F760   E3F2C140 40404040 0000050C 00000046   *. ....a.328.607-T2A     ........*
000020  F100F0F0 F0F00000 60F0F04B F0F0F060   F0F00000 00000000 C6C4F0C1 C3F2F9F2   *1.0000..-00.000-00......FD0AC292*

RECORD SEQUENCE NUMBER - 2
000000  00400010 003C8110 F3F2F84B F6F0F760   E3F2C140 40404040 0000050C 00000000   *. ....a.328.607-T2A     ........*
000020  F200F0F0 F0F00000 60F0F04B F0F0F560   F0F00000 00000000 C3C1F0F2 40404040   *2.0000..-00.005-00......CA02    *

RECORD SEQUENCE NUMBER - 3
000000  00400010 003C8110 F3F2F84B F6F1F360   E3F2C140 40404040 0000050C 00000045   *. ....a.328.613-T2A     ........*
000020  F100F0F0 F0F00000 60F0F14B F1F7F060   F0F00000 00000000 C6C4F0C1 C3F2F9F2   *1.0000..-01.170-00......FD0AC292*

RECORD SEQUENCE NUMBER - 4
000000  00400010 003C8110 F3F2F94B F7F8F460   E3F2C140 40404040 60C1C4F4 40404040   *. ....a.329.784-T2A     -AD4    *
000020  C100F0F0 F0F00000 60F0F04B F0F0F060   F0F05100 00000000 C3C1F0F2 40404040   *A.0000..-00.000-00......CA02    *

RECORD SEQUENCE NUMBER - 5
000000  00400010 003C8110 F3F2F94B F7F8F560   E3F2C140 40404040 60C1C4F4 40404040   *. ....a.329.785-T2A     -AD4    *
000020  D700F0F0 F0F00000 60F0F04B F0F0F160   F0F05A00 00000000 C5D2C1D9 40404040   *P.0000..-00.001-00!.....EKAR    *

RECORD SEQUENCE NUMBER - 6
000000  008D0010 003C8110 F3F2F94B F7F8F660   E3F2C140 40404040 60C1C4F4 40404040   *.—....a.329.786-T2A     -AD4    *
000020  E200F0F1 F0F00000 60F0F04B F0F0F060   F0F05A00 00000000 00000000 00000000   *S.0100..-00.000-00!.............*
000040  004D0100 001C1001 0008A100 C3F4F1F0   0009A101 E2C4C6C3 F10007A1 02C1F2C1   *.(....... .C410.. .SDFC1.. .A2A*
000060  00141101 0009A101 E2C4C6C3 F20007A1   02C1F1C1 00141403 0008A201 E7F1F2C1   *...... .SDFC2.. .A1A......s.X12A*
000080  0008A202 E7F2F1C1 0005B004 E3                                             *..s.X21A..[.T                   *

RECORD SEQUENCE NUMBER - 7
000000  00440010 003C8110 F3F2F94B F7F8F660   E3F2C140 40404040 60C1C4F4 40404040   *......a.329.786-T2A     -AD4    *
000020  E500F8F1 C6C60000 60F0F04B F1F8F560   F0F05A00 00000000 00000000 00000000   *V.81FF..-00.185-00!.............*
000040  000481FF                                                                  *..a.                            *

RECORD SEQUENCE NUMBER - 8
000000  00C00010 003C8110 F3F3F04B F1F2F960   E3F2C140 40404040 60C1C4F4 40404040   *.{....a.330.129-T2A     -AD4    *
000020  E200F0F1 F0F20000 60F0F04B F0F0F060   F0F05A00 00000000 00000000 00000000   *S.0102..-00.000-00!.............*
000040  00800102 001C1001 0008A100 C3F4F1F0   0009A101 E2C4C6C3 F10007A1 02C1F2C1   *.......... .C410.. .SDFC1.. .A2A*
000060  00141101 0009A101 E2C4C6C3 F20007A1   02C1F1C1 00149201 C2F1F8F2 C1F1F6C1   *...... .SDFC2.. .A1A..k.B182A16A*
000080  C4F4C3C4 F1F8F0F3 00141403 0008A201   E7F1F2C1 0008A202 E7F2F1C1 00109301   *D4CD1803......s.X12A..s.X21A..l.*
0000A0  F9F8F1F2 F1F6F1F1 F4F4F4F3 0005B001   F00005B0 02D30005 B000D500 05B003F2   *981216114443..[.0..[.L..[.N..[.2*

RECORD SEQUENCE NUMBER - 9
000000  014E0010 003C8110 F3F3F04B F1F2F960   E3F2C140 40404040 60C1C4F4 40404040   *.+....a.330.129-T2A     -AD4    *
000020  E200F0F1 F2F00000 60F0F04B F0F0F060   F0F05A00 00000012 00000000 00000000   *S.0120..-00.000-00!.............*
000040  01020120 00401002 002CA001 C1D7D7C3   40C3D6D5 D5C5C3E3 C9D6D540 E2C9C6C9   *..... ....È.APPC CONNECTION SIFI*
000060  40C3C9C3 E2F140E3 D640E2C9 C6C940C3   C9C3E2F2 0009A101 E2C4C6C3 F10007A1   * CICS1 TO SIFI CICS2.. .SDFC1.. *
000080  02C1F2C1 00141102 0009A101 E2C4C6C3   F20007A1 02C1F1C1 001C9601 B182A16A   *.A2A...... .SDFC2.. .A1A..o.¶b .*
0000A0  D0519000 B1F4B7B2 DACB9711 AFB96CB1   F1359152 000C9600 B182A16A D06B6C00   *}...¶4»ý..p..5%¶1.j...o.¶b .},%.*
0000C0  00089204 F0F0F5F8 00079604 F0F0F100   0C9203B1 82A16ABE DCAE0100 379609E3   *..k.0058..o.001..k.¶b .Ø.õ...o.T*
0000E0  85A2A340 9485A2A2 81878540 F1408696   99408396 95A58599 A281A389 969540A3   *est message 1 for conversation t*
000100  99818385 40849683 A4948595 A381A389   96950014 9202C2F1 F8F2C1F1 F6C1C2C5   *race documentation..k.B182A16ABE*
000120  C3F6F7F0 F0F10005 B000D500 059602C4   00089603 C4C5D4D6 0005B001 F10005B0   *C67001..[.N..o.D..o.DEMO..[.1..[*
000140  02D3000C 81210000 00000000 0012                                           *.L..a.........                  *

RECORD SEQUENCE NUMBER - 10
000000  005A0010 003C8110 F3F3F04B F1F3F060   E3F2C140 40404040 60C1C4F4 40404040   *.!....a.330.130-T2A     -AD4    *
000020  E200F8F1 F2F30000 60F0F04B F0F0F060   F0F05A00 00000000 00000000 00000000   *S.8123..-00.000-00!.............*
000040  001A8123 00000000 BA49F9E5 721ED5D2   D0BB07B7 1A810D72 BC0D                  *..a.......9V..NK}Ù.».a..¿.      *

RECORD SEQUENCE NUMBER - 11
000000  00440010 003C8110 F3F3F04B F1F3F060   E3F2C140 40404040 60C1C4F4 40404040   *......a.330.130-T2A     -AD4    *
000020  E300F8F1 C6C60000 60F0F04B F0F0F360   F0F05A00 00000000 00000000 00000000   *T.81FF..-00.003-00!.............*
000040  000481FF                                                                  *..a.                            *

RECORD SEQUENCE NUMBER - 12
000000  00C00010 003C8110 F3F3F04B F2F1F560   E3F2C140 40404040 60C1C4F4 40404040   *.{....a.330.215-T2A     -AD4    *
000020  E200F0F1 F0F20000 60F0F04B F0F0F160   F0F05A00 00000000 00000000 00000000   *S.0102..-00.001-00!.............*
000040  00800102 001C1001 0008A100 C3F4F1F0   0009A101 E2C4C6C3 F10007A1 02C1F2C1   *.......... .C410.. .SDFC1.. .A2A*
000060  00141101 0009A101 E2C4C6C3 F20007A1   02C1F1C1 00149201 C2F1F8F2 C1F1F6C1   *...... .SDFC2.. .A1A..k.B182A16A*
000080  C5F9C3F7 F0F0F0F2 00141403 0008A201   E7F1F2C1 0008A202 E7F2F1C1 00109301   *E9C70002......s.X12A..s.X21A..l.*
0000A0  F9F8F1F2 F1F6F1F1 F4F4F4F3 0005B001   F00005B0 02D30005 B000D500 05B003F2   *981216114443..[.0..[.L..[.N..[.2*
```

```
RECORD SEQUENCE NUMBER - 13
000000  014E0010 003C8110 F3F3F04B F2F1F760   E3F2C140 40404040 60C1C4F4 40404040   *.+....a.330.217-T2A     -AD4    *
000020  E200F0F1 F2F00000 60F0F04B F0F0F060   F0F05A00 00000012 00000000 00000000   *S.0120..-00.000-00!.............*
000040  01020120 00401002 002CA001 C1D7D7C3   40C3D6D5 D5C5C3E3 C9D6D540 E2C9C6C9   *..... ....È.APPC CONNEÇTION SIFI̱*
000060  40C3C9C3 E2F140E3 D640E2C9 C6C940C3   C9C3E2F2 0009A101 E2C4C6C3 F10007A1   * CICS1 TO ŞIFI CICS2̰.. ˜.SDFC1.̰ ˜*
000080  02C1F2C1 00141102 0009A101 E2C4C6C3   F20007A1 02C1F1C1 001C9601 B182A16A   *.A2A...... ˜.SDFC2.. ˜.A1A..o̰.¶b ˜.*
0000A0  E0A6DC00 4D5DBA09 9DA6B495 5923C884   E5C59A00 000C9600 B182A16A E0BFCC00   *.w..()..″w╕n..HdVE...̰.o.¶b ˜..Å..*
0000C0  00089204 F0F0F5F9 00079604 F0F0F200   0C9203B1 82A16ADA 4A520100 379609E3   *..k.0059..o.002..k.¶b ˜..>....o.T*
0000E0  85A2A340 9485A2A2 81878540 F2408696   99408396 95A58599 A281A389 969540A3   *est message 2 for conversation t*
000100  99818385 40849683 A4948595 A381A389   96950014 9202C2F1 F8F2C1F1 F6C1C4C1   *race documentation..k.B182A16ADA*
000120  F2C5F5F8 F0F10005 B000D500 059602C4   00089603 C4C5D4D6 0005B001 F10005B0   *2E5801..[.N..o.D..o.DEMO..[.1..[*
000140  02D3000C 81210000 00000000 0012                                             *.L..a.........                  *

RECORD SEQUENCE NUMBER - 14
000000  005A0010 003C8110 F3F3F04B F2F1F760   E3F2C140 40404040 60C1C4F4 40404040   *.!....a.330.217-T2A     -AD4    *
000020  E200F8F1 F2F30000 60F0F04B F0F0F060   F0F05A00 00000000 00000000 00000000   *S.8123..-00.000-00!.............*
000040  001A8123 00000000 894909D1 AFDB9FD5   84BD3CFC A7DCB380 3426                *..a.....i..J..5Nd]..x.....      *
RECORD SEQUENCE NUMBER - 15
000000  00440010 003C8110 F3F3F04B F2F1F860   E3F2C140 40404040 60C1C4F4 40404040   *......a.330.218-T2A     -AD4    *
000020  E300F8F1 C6C60000 60F0F04B F0F0F360   F0F05A00 00000000 00000000 00000000   *T.81FF..-00.003-00!.............*
000040  000481FF                                                                    *..a.                            *

RECORD SEQUENCE NUMBER - 16
000000  00400010 003C8110 F3F3F04B F2F3F560   E3F2C140 40404040 60C1C4F4 40404040   *. ....a.330.235-T2A     -AD4    *
000020  E500F0F0 F0F00000 60F0F04B F0F7F060   F0F05A00 00000000 00000000 00000000   *V.0000..-00.070-00!.............*

RECORD SEQUENCE NUMBER - 17
000000  00400010 003C8110 F3F3F04B F3F8F960   E3F2C140 40404040 60C1C4F4 40404040   *. ....a.330.389-T2A     -AD4    *
000020  C600F0F0 F0F00000 60F0F04B F0F0F460   F0F00000 00000000 00000000 00000000   *F.0000..-00.004-00..............*
```

In this example:

- Record 1 shows that there was no connection to the partner system (netname FD0AC292) when the MTP was started (the CICS connection status CVDA is 46).
- Record 2 shows that the MTP is customized to acquire a connection immediately.
- Record 3 shows that the connection was successfully established after less than 2 seconds.
- Records 4 and 5 show the CICS **allocate session** and **connect process** commands. The partner system identifier is CA02 and the partner process identifier is EKAR.
- Records 6 and 7 show the PROBE PDU that was sent to the partner process with a request for confirmation.
- Records 8 to 11 contain a trace of the first message sent to the partner process. The second message is traced in records 12 to 15.
- In this example, the MIP window size is 010, but only two messages were sent, so the message transfer confirmation had to be requested in a separate command. Record 16 shows the request for confirmation that was sent in a **send confirm** command without data.
- Record 17 shows the **free** command, which deallocates the session to the partner system.

The timestamps in the conversation trace records indicate when the conversation commands were issued. From these timestamps and the timestamps in the example of a weak external conversation trace shown in "Example 3: Receive Two Application Messages" on page 143, you can calculate the time required for the flow of application data and APPC control information between the partner systems. In this example:

- The first message was prepared within 158 milliseconds (from 329.971 to 330.129). The four **send** commands to send the first message were issued within 1 millisecond (330.129 - 330.130). Data transmission (flush VTAM buffer) was requested after a **send** command for a PDU Trailer (see records 4 to 7 in "Example 3: Receive Two Application Messages" on page 143).

- The second message was prepared within the next 82 milliseconds (from 330.133 to 330.215), and the four **send** commands for the second message were issued within 3 milliseconds.
- It took MERVA Link 14 milliseconds to realize that the last message had been sent, and that a confirmation for the two messages needed to be requested. The confirmation was received after 70 milliseconds, and the two confirmed messages were processed by MERVA Link within 84 milliseconds (330.305 to 330.389).

Example 2: Receive Two Acknowledgments

The following is an example of a full external conversation trace written by a receiving MTP during the transfer of two acknowledgment messages:

```
RECORD SEQUENCE NUMBER - 18
000000  008D0010 003C8110 F4F0F64B F9F9F260   C3C1F0F2 40404040 60C1C4F7 40404040   *.-....a.406.992-CA02    -AD7    *
000020  D910F0F1 F0F00006 60F0F04B F0F0F060   F1F00000 00000000 00200000 00000000   *R.0100..-00.000-10..............*
000040  004D0100 001C1001 0008A100 C3F4F1F0   0009A101 E2C4C6C3 F20007A1 02C1F1C1   *.(....... .C410.. .SDFC2.. .A1A*
000060  00141101 0009A101 E2C4C6C3 F10007A1   02C1F2C1 00141403 0008A201 E7F2F1C1   *...... .SDFC1.. .A2A......s.X21A*
000080  0008A202 E7F1F2C1 0005B004 E3                                                *..s.X12A..[.T                  *

RECORD SEQUENCE NUMBER - 19
000000  00440010 003C8110 F4F0F64B F9F9F360   C3C1F0F2 40404040 60C1C4F7 40404040   *......a.406.993-CA02    -AD7    *
000020  D918F8F1 C6C60006 60F0F04B F0F0F060   F1F80000 00000000 00200000 00000000   *R.81FF..-00.000-18..............*
000040  000481FF                                                                     *..a.                            *

RECORD SEQUENCE NUMBER - 20
000000  00400010 003C8110 F4F0F74B F0F4F360   C3C1F0F2 40404040 60C1C4F7 40404040   *. ....a.407.043-CA02    -AD7    *
000020  C300F0F0 F0F00000 60F0F04B F0F0F260   F0F00000 00000000 00000000 00000000   *C.0000..-00.002-00..............*

RECORD SEQUENCE NUMBER - 21
000000  00C00010 003C8110 F4F0F74B F0F4F660   C3C1F0F2 40404040 60C1C4F7 40404040   *.{....a.407.046-CA02    -AD7    *
000020  D910F0F1 F0F20006 60F0F04B F0F7F060   F1F00000 00000000 00200000 00000000   *R.0102..-00.070-10..............*
000040  00800102 001C1001 0008A100 C3F4F1F0   0009A101 E2C4C6C3 F20007A1 02C1F1C1   *.......... .C410.. .SDFC2.. .A1A*
000060  00141101 0009A101 E2C4C6C3 F10007A1   02C1F2C1 00149201 C2F1F8F2 C1F1C2F4   *...... .SDFC1.. .A2A..k.B182A1B4*
000080  F3C5C5C4 F9C1F0F7 00141403 0008A201   E7F2F1C1 0008A202 E7F1F2C1 00109301   *3EED9A07......s.X21A..s.X12A..l.*
0000A0  F9F8F1F2 F1F6F1F1 F4F6F0F0 0005B001   F00005B0 02C80005 B0004000 05B003F2   *981216114600..[.0..[.H..[. ..[.2*

RECORD SEQUENCE NUMBER - 22
000000  00CF0010 003C8110 F4F0F74B F1F1F660   C3C1F0F2 40404040 60C1C4F7 40404040   *......a.407.116-CA02    -AD7    *
000020  D910F0F1 F1F20006 60F0F04B F0F0F060   F1F00000 00000000 00200000 00000000   *R.0112..-00.000-10..............*
000040  008F0112 00141102 0009A101 E2C4C6C3   F20007A1 02C1F1C1 00149202 C2F1F8F2   *.Å........ .SDFC2.. .A1A..k.B182*
000060  C1F1F6C1 C2C5C3F6 F7F0F0F1 00089204   F3F2F6F5 00079604 F0F0F100 0C9203B1   *A16ABEC67001..k.3265..o.001..k.¶*
000080  82A1B433 85BA0000 48150000 109301F9   F8F1F2F1 F6F1F1F4 F5F5F900 069501F0   *b ┐.e.......l.981216114559..n.0*
0000A0  F0000A95 02D6D240 40404000 249503D4   85A2A281 878540F1 40A2A483 8385A2A2   *0..n.OK    ..n.Message 1 success*
0000C0  86A49393 A8409799 968385A2 A28584                                            *fully processed                *

RECORD SEQUENCE NUMBER - 23
000000  00440010 003C8110 F4F0F74B F1F1F760   C3C1F0F2 40404040 60C1C4F7 40404040   *......a.407.117-CA02    -AD7    *
000020  D918F8F1 C6C60006 60F0F04B F0F0F060   F1F80000 00000000 00200000 00000000   *R.81FF..-00.000-18..............*
000040  000481FF                                                                     *..a.                            *

RECORD SEQUENCE NUMBER - 24
000000  00400010 003C8110 F4F0F74B F2F0F460   C3C1F0F2 40404040 60C1C4F7 40404040   *. ....a.407.204-CA02    -AD7    *
000020  C300F0F0 F0F00000 60F0F04B F0F0F060   F0F00000 00000000 00000000 00000000   *C.0000..-00.000-00..............*

RECORD SEQUENCE NUMBER - 25
000000  00C00010 003C8110 F4F0F74B F2F0F660   C3C1F0F2 40404040 60C1C4F7 40404040   *.{....a.407.206-CA02    -AD7    *
000020  D910F0F1 F0F20006 60F0F04B F1F2F460   F1F00000 00000000 00200000 00000000   *R.0102..-00.124-10..............*
000040  00800102 001C1001 0008A100 C3F4F1F0   0009A101 E2C4C6C3 F20007A1 02C1F1C1   *.......... .C410.. .SDFC2.. .A1A*
000060  00141101 0009A101 E2C4C6C3 F10007A1   02C1F2C1 00149201 C2F1F8F2 C1F1C2F4   *...... .SDFC1.. .A2A..k.B182A1B4*
000080  F7F3C1C5 F9C3F0F8 00141403 0008A201   E7F2F1C1 0008A202 E7F1F2C1 00109301   *73AE9C08......s.X21A..s.X12A..l.*
0000A0  F9F8F1F2 F1F6F1F1 F4F6F0F0 0005B001   F00005B0 02C80005 B0004000 05B003F2   *981216114600..[.0..[.H..[. ..[.2*

RECORD SEQUENCE NUMBER - 26
000000  00CF0010 003C8110 F4F0F74B F3F3F060   C3C1F0F2 40404040 60C1C4F7 40404040   *......a.407.330-CA02    -AD7    *
000020  D910F0F1 F1F20006 60F0F04B F0F0F060   F1F00000 00000000 00200000 00000000   *R.0112..-00.000-10..............*
000040  008F0112 00141102 0009A101 E2C4C6C3   F20007A1 02C1F1C1 00149202 C2F1F8F2   *.Å........ .SDFC2.. .A1A..k.B182*
000060  C1F1F6C1 C4C1F2C5 F5F8F0F1 00089204   F3F2F6F6 00079604 F0F0F100 0C9203B1   *A16ADA2E5801..k.3266..o.001..k.¶*
000080  82A1B46A 69720600 48150000 109301F9   F8F1F2F1 F6F1F1F4 F5F5F900 069501F0   *b ┐..........l.981216114559..n.0*
0000A0  F0000A95 02D6D240 40404000 249503D4   85A2A281 878540F2 40A2A483 8385A2A2   *0..n.OK    ..n.Message 2 success*
0000C0  86A49393 A8409799 968385A2 A28584                                            *fully processed                *
```

```
RECORD SEQUENCE NUMBER - 27
000000  00440010 003C8110 F4F0F74B F3F3F160   C3C1F0F2 40404040 60C1C4F7 40404040   *......a.407.331-CA02    -AD7    *
000020  D918F8F1 C6C60006 60F0F04B F0F0F060   F1F80000 00000000 00200000 00000000   *R.81FF..-00.000-18.............*
000040  000481FF                                                                    *..a.                            *

RECORD SEQUENCE NUMBER - 28
000000  00400010 003C8110 F4F0F74B F3F8F760   C3C1F0F2 40404040 60C1C4F7 40404040   *. ....a.407.387-CA02    -AD7    *
000020  C300F0F0 F0F00000 60F0F04B F0F0F160   F0F00000 00000000 00000000 00000000   *C.0000..-00.001-00.............*

RECORD SEQUENCE NUMBER - 29
000000  00400010 003C8110 F4F0F74B F3F8F860   C3C1F0F2 40404040 60C1C4F7 40404040   *. ....a.407.388-CA02    -AD7    *
000020  D903F0F0 F0F00006 60F0F04B F0F8F360   F0F30000 00000000 00200000 00000000   *R.0000..-00.083-03.............*

RECORD SEQUENCE NUMBER - 30
000000  00400010 003C8110 F4F0F74B F4F7F260   C3C1F0F2 40404040 60C1C4F7 40404040   *. ....a.407.472-CA02    -AD7    *
000020  C600F0F0 F0F00000 60F0F04B F0F0F260   F0F00000 00000000 00000000 00000000   *F.0000..-00.002-00.............*
```

In this example:

- The MIP window size of the sending ASP is 001.
- Records 18 and 19 show the PROBE PDU that is received from the partner process with a request for confirmation (bit X'08' of the compressed conversation indicator in record 19 is set).
- Record 20 shows the requested confirmation.
- Records 21 to 23 contain a trace of the first acknowledgment message received from the partner process.
- Record 24 shows the requested confirmation.
- Records 25 to 27 contain a trace of the second acknowledgment message.
- Record 28 shows the requested confirmation.
- Record 29 shows the trace of another **receive** command. The compressed conversation indicators (X'03') indicate that no data has been received (X'02'), and that the receiving process must terminate the conversation (X'01').
- Record 30 shows the required **free** command.
- Records 24 and 28 show the transfer confirmation that was requested when the PDU trailer of each of the acknowledgment message PDUs was received. Note that the MIP window size at CICS 2, the sender of the acknowledgment messages, is 001. This means, a transfer confirmation is requested for each message.

Weak External Conversation Trace Samples

These examples of weak conversation traces were generated by MERVA Link transactions in CICS 2.

Example 3: Receive Two Application Messages

The following is an example of a weak external conversation trace written by a receiving MTP during the transfer of two application messages:

```
RECORD SEQUENCE NUMBER - 1
000000  00400010 003C8110 F3F2F94B F8F2F860   C3C1F0F1 40404040 60C1C2E3 40404040   *. ....a.329.828-CA01    -ABT    *
000020  D910F0F1 F0F00006 60F0F04B F0F0F060   F1F00000 00000000 00200000 00000000   *R.0100..-00.000-10.............*

RECORD SEQUENCE NUMBER - 2
000000  00400010 003C8110 F3F2F94B F8F2F960   C3C1F0F1 40404040 60C1C2E3 40404040   *. ....a.329.829-CA01    -ABT    *
000020  D918F8F1 C6C60006 60F0F04B F0F0F060   F1F80000 00000000 00200000 00000000   *R.81FF..-00.000-18.............*

RECORD SEQUENCE NUMBER - 3
000000  00400010 003C8110 F3F2F94B F9F6F560   C3C1F0F1 40404040 60C1C2E3 40404040   *. ....a.329.965-CA01    -ABT    *
000020  C300F0F0 F0F00000 60F0F04B F0F0F360   F0F00000 00000000 00000000 00000000   *C.0000..-00.003-00.............*

RECORD SEQUENCE NUMBER - 4
000000  00400010 003C8110 F3F2F94B F9F6F860   C3C1F0F1 40404040 60C1C2E3 40404040   *. ....a.329.968-CA01    -ABT    *
000020  D910F0F1 F0F20006 60F0F04B F1F6F660   F1F00000 00000000 00200000 00000000   *R.0102..-00.166-10.............*

RECORD SEQUENCE NUMBER - 5
000000  00400010 003C8110 F3F3F04B F1F3F560   C3C1F0F1 40404040 60C1C2E3 40404040   *. ....a.330.135-CA01    -ABT    *
000020  D910F0F1 F2F00006 60F0F04B F0F0F060   F1F00000 00000012 00200000 00000000   *R.0120..-00.000-10.............*

RECORD SEQUENCE NUMBER - 6
000000  00400010 003C8110 F3F3F04B F1F3F660   C3C1F0F1 40404040 60C1C2E3 40404040   *. ....a.330.136-CA01    -ABT    *
000020  D910F8F1 F2F30006 60F0F04B F0F0F060   F1F00000 00000000 00200000 00000000   *R.8123..-00.000-10.............*

RECORD SEQUENCE NUMBER - 7
000000  00400010 003C8110 F3F3F04B F1F3F660   C3C1F0F1 40404040 60C1C2E3 40404040   *. ....a.330.136-CA01    -ABT    *
000020  D910F8F1 C6C60006 60F0F04B F0F0F060   F1F00000 00000000 00200000 00000000   *R.81FF..-00.000-10.............*

RECORD SEQUENCE NUMBER - 8
000000  00400010 003C8110 F3F3F04B F2F5F160   C3C1F0F1 40404040 60C1C2E3 40404040   *. ....a.330.251-CA01    -ABT    *
000020  D910F0F1 F0F20006 60F0F04B F0F0F360   F1F00000 00000000 00200000 00000000   *R.0102..-00.003-10.............*

RECORD SEQUENCE NUMBER - 9
000000  00400010 003C8110 F3F3F04B F2F5F560   C3C1F0F1 40404040 60C1C2E3 40404040   *. ....a.330.255-CA01    -ABT    *
000020  D910F0F1 F2F00006 60F0F04B F0F0F060   F1F00000 00000012 00200000 00000000   *R.0120..-00.000-10.............*

RECORD SEQUENCE NUMBER - 10
000000  00400010 003C8110 F3F3F04B F2F5F660   C3C1F0F1 40404040 60C1C2E3 40404040   *. ....a.330.256-CA01    -ABT    *
000020  D910F8F1 F2F30006 60F0F04B F0F0F060   F1F00000 00000000 00200000 00000000   *R.8123..-00.000-10.............*

RECORD SEQUENCE NUMBER - 11
000000  00400010 003C8110 F3F3F04B F2F5F660   C3C1F0F1 40404040 60C1C2E3 40404040   *. ....a.330.256-CA01    -ABT    *
000020  D918F8F1 C6C60006 60F0F04B F0F0F060   F1F80000 00000000 00200000 00000000   *R.81FF..-00.000-18.............*

RECORD SEQUENCE NUMBER - 12
000000  00400010 003C8110 F3F3F04B F3F0F260   C3C1F0F1 40404040 60C1C2E3 40404040   *. ....a.330.302-CA01    -ABT    *
000020  C300F0F0 F0F00000 60F0F04B F0F0F160   F0F00000 00000000 00000000 00000000   *C.0000..-00.001-00.............*

RECORD SEQUENCE NUMBER - 13
000000  00400010 003C8110 F3F3F04B F3F0F460   C3C1F0F1 40404040 60C1C2E3 40404040   *. ....a.330.304-CA01    -ABT    *
000020  D903F0F0 F0F00006 60F0F04B F0F8F960   F0F30000 00000000 00200000 00000000   *R.0000..-00.089-03.............*

RECORD SEQUENCE NUMBER - 14
000000  00400010 003C8110 F3F3F04B F3F9F460   C3C1F0F1 40404040 60C1C2E3 40404040   *. ....a.330.394-CA01    -ABT    *
000020  C600F0F0 F0F00000 60F0F04B F0F0F060   F0F00000 00000000 00000000 00000000   *F.0000..-00.000-00.............*
```

In this example:

- When the receiving process confirmed the PROBE (record 3), it issued a **receive** command at 329.968 seconds (record 4). The sending partner process received that confirmation and sent the first message within 166 milliseconds.
- The complete first message was received by the receiving process 3 milliseconds after the sending process sent the PDU Trailer (records 4 to 7). The first message was delivered to the receiving application within the next 115 milliseconds, and the second message was delivered within 46 milliseconds (see records 11 and 12).
- The request for confirmation for the two messages is received together with the PDU trailer of the second message (record 11). The compressed conversation indicators (X'18') ask for confirmation (X'08') and indicate that another **receive** command must be issued after the requested **confirm** command (X'10').
- The request to terminate the conversation (X'01') was received as response to a **receive** command (record 13). This command did not return data (X'02').

This trace corresponds to the trace shown in “Example 1: Send Two Application Messages” on page 139.

Example 4: Send Two Acknowledgment Messages

The following is an example of a weak external conversation trace written by a sending MTP during the transfer of two acknowledgment messages:

```
RECORD SEQUENCE NUMBER - 15
000000  00400010 003C8110 F4F0F64B F9F4F760   E3F1C140 40404040 0000071C 00000045   *. ....a.406.947-T1A     ........*
000020  F100F0F0 F0F00000 60F0F04B F0F0F060   F0F00000 00000000 C6C4F0C1 C3F2F9F1   *1.0000..-00.000-00......FD0AC291*

RECORD SEQUENCE NUMBER - 16
000000  00400010 003C8110 F4F0F64B F9F4F860   E3F1C140 40404040 60C1C2D8 40404040   *. ....a.406.948-T1A     -ABQ    *
000020  C100F0F0 F0F00000 60F0F04B F0F0F060   F0F05100 00000000 C3C1F0F1 40404040   *A.0000..-00.000-00......CA01    *

RECORD SEQUENCE NUMBER - 17
000000  00400010 003C8110 F4F0F64B F9F4F860   E3F1C140 40404040 60C1C2D8 40404040   *. ....a.406.948-T1A     -ABQ    *
000020  D700F0F0 F0F00000 60F0F04B F0F0F160   F0F05A00 00000000 C5D2C1D9 40404040   *P.0000..-00.001-00!.....EKAR    *

RECORD SEQUENCE NUMBER - 18
000000  00400010 003C8110 F4F0F64B F9F5F060   E3F1C140 40404040 60C1C2D8 40404040   *. ....a.406.950-T1A     -ABQ    *
000020  E200F0F1 F0F00000 60F0F04B F0F0F060   F0F05A00 00000000 00000000 00000000   *S.0100..-00.000-00!............*

RECORD SEQUENCE NUMBER - 19
000000  00400010 003C8110 F4F0F64B F9F5F060   E3F1C140 40404040 60C1C2D8 40404040   *. ....a.406.950-T1A     -ABQ    *
000020  E500F8F1 C6C60000 60F0F04B F0F9F860   F0F05A00 00000000 00000000 00000000   *V.81FF..-00.098-00!............*

RECORD SEQUENCE NUMBER - 20
000000  00400010 003C8110 F4F0F74B F1F1F060   E3F1C140 40404040 60C1C2D8 40404040   *. ....a.407.110-T1A     -ABQ    *
000020  E200F0F1 F0F20000 60F0F04B F0F0F060   F0F05A00 00000000 00000000 00000000   *S.0102..-00.000-00!............*

RECORD SEQUENCE NUMBER - 21
000000  00400010 003C8110 F4F0F74B F1F1F060   E3F1C140 40404040 60C1C2D8 40404040   *. ....a.407.110-T1A     -ABQ    *
000020  E200F0F1 F1F20000 60F0F04B F0F0F060   F0F05A00 00000000 00000000 00000000   *S.0112..-00.000-00!............*

RECORD SEQUENCE NUMBER - 22
000000  00400010 003C8110 F4F0F74B F1F1F160   E3F1C140 40404040 60C1C2D8 40404040   *. ....a.407.111-T1A     -ABQ    *
000020  E500F8F1 C6C60000 60F0F04B F0F9F760   F0F05A00 00000000 00000000 00000000   *V.81FF..-00.097-00!............*

RECORD SEQUENCE NUMBER - 23
000000  00400010 003C8110 F4F0F74B F3F2F660   E3F1C140 40404040 60C1C2D8 40404040   *. ....a.407.326-T1A     -ABQ    *
000020  E200F0F1 F0F20000 60F0F04B F0F0F060   F0F05A00 00000000 00000000 00000000   *S.0102..-00.000-00!............*

RECORD SEQUENCE NUMBER - 24
000000  00400010 003C8110 F4F0F74B F3F2F660   E3F1C140 40404040 60C1C2D8 40404040   *. ....a.407.326-T1A     -ABQ    *
000020  E200F0F1 F1F20000 60F0F04B F0F0F060   F0F05A00 00000000 00000000 00000000   *S.0112..-00.000-00!............*

RECORD SEQUENCE NUMBER - 25
000000  00400010 003C8110 F4F0F74B F3F2F760   E3F1C140 40404040 60C1C2D8 40404040   *. ....a.407.327-T1A     -ABQ    *
000020  E500F8F1 C6C60000 60F0F04B F0F6F360   F0F05A00 00000000 00000000 00000000   *V.81FF..-00.063-00!............*

RECORD SEQUENCE NUMBER - 26
000000  00400010 003C8110 F4F0F74B F4F6F960   E3F1C140 40404040 60C1C2D8 40404040   *. ....a.407.469-T1A     -ABQ    *
000020  C600F0F0 F0F00000 60F0F04B F0F0F460   F0F00000 00000000 00000000 00000000   *F.0000..-00.004-00.............*
```

In this example:

- The MIP window size was 001.
- The connection to the partner system was immediately available when the MTP starts (record 15).
- A confirmation was requested for the initial probe (record 19) as well as for each message (records 22 and 25).

This trace corresponds to the trace shown in “Example 2: Receive Two Acknowledgments” on page 142.

The Internal Conversation Trace

The MERVA Link internal conversation trace is written by a sending and by a receiving message transfer program to the internal conversation trace area of that program. The internal conversation trace area is a part of a larger main storage area, the message transfer program (permanent) work area. The length of the internal conversation trace area is individually specified by each MERVA Link message transfer program. This area provides space for a specific maximum number of trace entries. It is used in wrap-around mode. The last trace entry is identified by the timestamp in all entries.

The MERVA Link internal conversation trace, which is supported in all MERVA Link CICS environments and in the MERVA Link IMS APPC environment (APPC/MVS) as a weak trace (this means, no application data is traced), is always written. It cannot be disabled. The control information contained in an internal conversation trace entry and the layout of that entry is individually specified by each MERVA Link message transfer program.

The internal conversation trace contains only communication commands issued in one task. A trace entry is written as soon as control is returned to the MTP after issuing the corresponding APPC command.

The MERVA Link message transfer programs that write an internal conversation trace are listed in Table 6.

Table 6. MERVA Link Message Transfer Programs

MTP Name	Description
EKATS10	A sending MTP that supports APPC in the MERVA ESA CICS environment.
EKATR10	A receiving MTP that supports APPC in the MERVA ESA CICS environment.
EKATPO1	A sending MTP that supports APPC/MVS in the MERVA ESA IMS environment.
EKATPI1	A receiving MTP that supports APPC/MVS and APPC/IMS in the MERVA ESA IMS environment.

The Back-to-Back TP Mirror EKATM10 does not issue APPC conversation commands, and therefore does not support an internal conversation trace.

EKATS10 Internal Conversation Trace

The length of the internal conversation trace area of the MERVA Link sending MTP in the CICS environment is 4096 bytes. It is identified by the eye catcher **EKATS10 INTERNAL CONV TRACE AREA** in a storage dump of a MERVA Link CICS sending task.

The length of a conversation trace entry is 64 bytes. The length, layout, and contents are the same as a weak external conversation trace entry (see Table 5 on page 139). The maximum number of entries in the EKATS100 conversation trace area is 64.

EKATR10 Internal Conversation Trace

The length of the internal conversation trace area of the MERVA Link sending MTP in the CICS environment is 4096 bytes. It is identified by the eye catcher **EKATR10 INTERNAL CONV TRACE AREA** in a storage dump of a MERVA Link CICS receiving task.

The length of a conversation trace entry is 64 bytes. The length, layout, and contents are the same as a weak external conversation trace entry (see Table 5 on page 139). The maximum number of entries in the EKATR10 conversation trace area is 64.

EKATPO1 Internal Conversation Trace

The length of the internal conversation trace area of the MERVA Link sending MTP in the IMS APPC/MVS environment is 2048 bytes. It is identified by the eye catcher **EKATPO1 INTERNAL CONV TRACE AREA** in a storage dump of a MERVA Link IMS APPC sending task.

The length of a conversation trace entry is 32 bytes. The maximum number of entries in the EKATPO1 conversation trace area is 64.

The format of a conversation trace entry, this is, the displacement of its fields in the entry, the field lengths, and a description of each field is shown in Table 7.

Table 7. EKATPO1 Conversation Trace Entry

Displ.	Length	Field Description
00	1	Command Identifier (see Table 3 on page 136)
01	1	Reserved
02	4	Application Data Identifier in characters
06	2	Bytes 2 and 3 of the ATB_RETURN_CODE
08	8	Relative Time in the format SSS.FFF- (seconds and milliseconds)
16	8	Command Duration in the format -SS.FFF- (seconds and milliseconds)
24	8	Diagnostic control information (optional)

Figure 9 shows a sample internal conversation trace written by EKATPO1 during the transfer of two application messages from APPC/MVS to CICS/ESA (R).

```
0008C7E0 C5D2C1E3 D7D6F140 C9D5E3C5 D9D5C1D3    40C3D6D5 E540E3D9 C1C3C540 C1D9C5C1   *EKATPO1 INTERNAL CONV TRACE AREA*
0008C800 C100F0F0 F0F00000 F4F5F94B F8F1F760    60F0F14B F5F1F860 00000000 00000000   *A.0000..459.817..01.518.........*
0008C820 E200F0F1 F0F00000 F4F6F14B F3F3F560    60F0F04B F1F0F660 00000000 00000000   *S.0100..461.335..00.106.........*
0008C840 E500F8F1 C6C60000 F4F6F14B F4F4F260    60F0F04B F4F1F160 00000000 00000000   *V.81FF..461.442..00.411.........*
0008C860 E200F0F1 F0F20000 F4F6F24B F0F2F560    60F0F04B F0F0F060 00000000 00000000   *S.0102..462.025..00.000.........*
0008C880 E200F0F1 F2F00000 F4F6F24B F0F2F660    60F0F04B F0F0F060 00000354 00000000   *S.0120..462.026..00.000.........*
0008C8A0 E200F8F1 F2F20000 F4F6F24B F0F2F760    60F0F04B F0F0F060 00000000 00000000   *S.8122..462.027..00.000.........*
0008C8C0 E300F8F1 C6C60000 F4F6F24B F0F2F860    60F0F04B F0F0F260 00000000 00000000   *T.81FF..462.028..00.002.........*
0008C8E0 E200F0F1 F0F20000 F4F6F24B F4F2F260    60F0F04B F0F0F160 00000000 00000000   *S.0102..462.422..00.001.........*
0008C900 E200F0F1 F2F00000 F4F6F24B F4F2F460    60F0F04B F0F0F060 00000354 00000000   *S.0120..462.424..00.000.........*
0008C920 E200F8F1 F2F20000 F4F6F24B F4F2F560    60F0F04B F0F0F060 00000000 00000000   *S.8122..462.425..00.000.........*
0008C940 E300F8F1 C6C60000 F4F6F24B F4F2F560    60F0F04B F0F0F460 00000000 00000000   *T.81FF..462.425..00.004.........*
0008C960 E500F0F0 F0F00000 F4F6F24B F5F0F260    60F0F04B F2F3F560 00000000 00000000   *V.0000..462.502..00.235.........*
0008C980 C600F0F0 F0F00000 F4F6F24B F9F6F360    60F0F04B F0F4F560 00000000 00000000   *F.0000..462.963..00.045.........*
```

Figure 9. EKATPO1 Conversation Trace: Send Two Application Messages

An **allocate** command in the APPC/MVS environment covers the functionality of both an **allocate** command and a **connect process** command in the CICS environment. That is why you cannot see an entry for a **connect process** command in conversation trace of program EKATPO1.

The diagnostic control information of the internal conversation trace records for an AMPDU Heading (PDU identifier X'0120') shows the total body data length within four bytes. It is X'00000354' in that example.

EKATPI1 Internal Conversation Trace

The length of the internal conversation trace area of the MERVA Link receiving MTP in the IMS APPC/MVS environment is 2048 bytes. It is identified by the eye catcher **EKATPI1 INTERNAL CONV TRACE AREA** in a storage dump of a MERVA Link APPC/MVS receiving task.

The length of a conversation trace entry is 32 bytes. The maximum number of entries in the EKATPI1 conversation trace area is 64.

The format of a conversation trace entry, that is, the displacement of its fields in the entry, the field lengths, and a description of each field is shown in Table 8.

Table 8. EKATPI1 Conversation Trace Entry

Displ.	Length	Field Description
00	1	Command Identifier (see Table 3 on page 136)
01	1	Compressed Conversation Indicators (see Table 4 on page 137)
02	4	Application Data Identifier in characters
06	1	Byte 3 of the ATB_RETURN_CODE
07	1	Byte 3 of the ATB Service Reason Code
08	8	Relative Time in the format SSS.FFF- (seconds and milliseconds)
16	8	Command Duration in the format -SS.FFF- (seconds and milliseconds)
24	8	Compressed conversation indicators in characters or additional control information (optional)

Figure 10 shows a sample internal conversation trace written by EKATPI1 during the transfer of two acknowledgment messages from CICS/ESA to APPC/MVS.

```
029057C0 C5D2C1E3 D7C9F140 C9D5E3C5 D9D5C1D3    40C3D6D5 E540E3D9 C1C3C540 C1D9C5C1   *EKATPI1 INTERNAL CONV TRACE AREA*
029057E0 D910F0F1 F0F00000 F2F5F74B F6F6F360    60F0F04B F0F0F060 F1F00000 00000000   *R.0100..257.663..00.000.10......*
02905800 D908F8F1 C6C60000 F2F5F74B F6F6F460    60F0F04B F0F0F060 F0F80000 00000000   *R.81FF..257.664..00.000.08......*
02905820 C310F0F0 F0F00000 F2F5F94B F7F9F660    60F0F04B F0F0F360 F1F00000 00000000   *C.0000..259.796..00.003.10......*
02905840 D910F0F1 F0F20000 F2F5F94B F8F0F060    60F0F04B F0F8F660 F1F00000 00000000   *R.0102..259.800..00.086.10......*
02905860 D910F0F1 F2F00000 F2F5F94B F8F8F660    60F0F04B F0F0F060 F1F00000 00000000   *R.0112..259.886..00.000.10......*
02905880 D910F8F1 C6C60000 F2F5F94B F8F8F760    60F0F04B F0F0F060 F1F00000 00000000   *R.81FF..259.887..00.000.10......*
029058A0 D910F0F1 F0F20000 F2F6F04B F0F6F960    60F0F04B F0F0F060 F1F00000 00000000   *R.0102..260.069..00.000.10......*
029058C0 D910F0F1 F2F00000 F2F6F04B F0F6F960    60F0F04B F0F0F060 F1F00000 00000000   *R.0112..260.069..00.000.10......*
029058E0 D908F8F1 C6C60000 F2F6F04B F0F6F960    60F0F04B F0F0F060 F1F00000 00000000   *R.81FF..260.069..00.000.08......*
02905900 C310F0F0 F0F00000 F2F6F04B F2F3F460    60F0F04B F0F0F360 F1F00000 00000000   *C.0000..260.234..00.003.10......*
02905920 D903F0F0 F0F00000 F2F6F04B F2F3F760    60F0F04B F1F0F360 F0F30000 00000000   *R.0000..260.237..00.103.03......*
02905940 C600F0F0 F0F00000 F2F6F04B F3F4F160    60F0F04B F0F3F660 F0F00000 00000000   *F.0000..260.341..00.036.00......*
```

Figure 10. EKATPI1 Conversation Trace: Receive Two Acknowledgments

Chapter 13. MERVA Link USS Problem Determination Aids

MERVA Link USS provides tracing and error reporting facilities for problem determination.

Processing Trace Facility

MERVA Link USS provides a processing trace facility that can be activated by setting values during customization, or by issuing MERVA Link application control commands. For an inbound conversation, the receiving process (ekatpi for SNA APPC or ekatci for TCP/IP) writes processing information to a trace file. For an outbound conversation, the sending ASP writes processing information to a trace file.

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

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

Each line in the trace file (a trace entry) begins with the name of the MERVA Link program that inserted the trace entry. The program name is followed by a 3-character identifier of the applicable internal function (for example, EKATPI_rds or EKAP1I_cip). The data of a trace entry is self explanatory. An excerpt from a routing process with trace level 1 is shown in Figure 11 on page 150.

```
EKATPI     Conversation start on Mon Oct 26 17:10:22 1998
EKATPI     Partner LU name is DEIBMFD.FD0AC291 , user ID is HUS
.
.
.
EKATPI_rcv Starting a new PDU   on Mon Oct 26 17:10:23 1998
EKATPI_rds PDU segment received is 004D 0100 , received length is 004D
EKATPI_rds PDU segment received is 0004 81FF , received length is 0004
EKATPI_rds Request for confirmation received
EKATPI_cmp Complete PDU rcvd    on Mon Oct 26 17:10:23 1998

EKATPI_hsi Partner MERVA system type/version C410 found in Probe PDU
.
.
.
EKAP1R_rtm Recipient node HUSX1 is not the local node
EKAP1R_rtm Route inbound PDU via TCP/IP at 623.170
EKAP1R_rtm Adjacent host name is husx.bs.boeblingen.ibm.com
.
.
.
EKATCO_con Service Primitive    is CONNECT.Request
EKATCO_con Partner host name    is husx.bs.boeblingen.ibm.com
EKATCO_con Partner host address is 9.164.170.51
EKATCO_con Partner port number  is 7110
EKATCO_con Probe Envelope start is 00710100 001C1001 0008A100

EKATCO_trm Returning from TCO with RC = 00 and RS = 00

EKAP1R_tme Activity complete    at 623.791, time delta is 000.621
EKAP1R_trm Returning from P1I with RC = 00 and RS = 00

EKATPI_val Transfer confirmation requested by partner TP

EKAP1I_cip Service Primitive    is ProcessPDU.Indication
EKAP1R_rtm Route request for confirmation via TCP/IP at 623.791

EKATCO_rtc Service Primitive    is SendPDU.Request
EKATCO_rtc Request confirmation without data

EKATCO_rci Request confirmation at 17:10:23 with 60 sec timeout
EKATCO_rci Return from read     at 17:10:24

EKATCO_trm Returning from TCO with RC = 00 and RS = 00

EKAP1R_tme Activity complete    at 624.131, time delta is 000.339
EKAP1R_trm Returning from P1I with RC = 00 and RS = 00

EKATPI_cfm Probe or message window confirmed

EKATPI_rcv Starting a new PDU   on Mon Oct 26 17:10:24 1998
EKATPI_rds Deallocated normal received
.
.
.
EKATPI     Conversation end   Mon Oct 26 17:10:24 1998
EKATPI     Elapsed time is 2 seconds
```

Figure 11. Routing Process Trace (SNA to TCP/IP)

Trace File Allocation Modes

There are two modes in which the MERVA Link USS processing trace facility can record trace data in files:

Sequential mode

The trace is written to a file with a name that includes a timestamp. Trace files are never overwritten.

Trace files have names of the form

trace_name*.t.*MMDDhhmmss.x

where:

trace_name	The name of the incoming SNA process, incoming TCP/IP process, or ASP that is writing the trace file
MMDDhhmmss	The month, day, hour, minute, and second the trace file is created
x	An index number that is set to 1 immediately after the generation of the ACT, incremented with each trace that is recorded, and reset to 1 when the MERVA Link daemon is started

Wrap-around mode

The trace is written to a file with a name that includes an index number. This number is set to 1 for the first trace, then incremented, up to a limit, for each new trace file. When the limit is reached, the number is reset to 1 and the process is repeated. Each time a number is reused, the old data in the file is overwritten.

Trace files have names of the form

trace_name*.trace.*n

where:

trace_name	The name of the incoming SNA process, incoming TCP/IP process, or ASP that is writing the trace file
n	The index number.

A different trace-file allocation mode can be specified for:

- Each sending ASP
- All receiving SNA APPC processes
- All receiving TCP/IP processes

Which mode is used is determined by the value of the trace-file wrap limit of the corresponding resource type:

- If the wrap limit is **0**, the sequential mode is used.
- If the wrap limit is **1 to 255**, the wrap-around mode is used. The trace-file wrap limit determines the maximum number of trace files.

Trace File Directory

The name of the directory to which a trace file is written is determined by a parameter of the MERVA Link USS application control table (ACT), which is described in “Application Control Table (ACT)” on page 97. Alternativey, you can use an ACC command to modify the directory path manually.

If a MERVA Link USS inbound TP process cannot attach to the ACT, and:

- The MERVA USS instance directory name is available, the process writes a trace to the **/trc** subdirectory of the instance directory (**$MERVA_DIR/trc**, for example **/u/merva1/trc**)
- The MERVA USS instance directory name is not available, the process writes a trace to the **/tmp** file system

If the request to open a trace file fails, an inbound TP process writes a trace file to the **/tmp** file system. This file has the name **ekatpi.trace** (for the SNA APPC inbound TP) or **ekatci.trace** (for the TCP/IP inbound TP), and gives the name of the trace file that could not be successfully opened, and the reason for the failure, for example, that the TP process had insufficient authorization to write to the trace directory.

If there is no more free space in the file system that contains the trace file directory, no trace is written. A directory entry might be generated, however the file size will be zero. A failure to write a trace does not influence the handling of messages, which is the main task of a MERVA Link process.

Trace Levels

The amount of information that is written to a trace file is specified by the trace level. A different trace level can be specified for:

- Each sending ASP
- All receiving SNA APPC processes
- All receiving TCP/IP processes

A trace level can be one of the following:

0 No trace is to be written. This is how you switch off tracing for a sending process or a group of receiving processes.

1 The process activity is to be traced.

2 The process activity and the transmitted control information (PDUs without the message text) is to be traced.

3 The process activity and the transmitted information (PDUs including the message text) is to be traced.

9 A time trace for performance analysis is to be conducted.

The MERVA Link USS trace facility can be completely disabled with the ACC command **rtd** (reset trace directory name).

Time Trace

The information contained in a MERVA Link processing trace file generated using trace level 9 (time trace) contains the following elements:

- For a send or receive message process:
  - Initial and final environment information of a processing trace at level 1
  - Service primitive identification entries at the P2 layer (entry)
  - Time trace entries at the P2 layer (exit)
- For a message routing process:
  - Initial and final environment information of a processing trace at level 1
  - Routing activity explanation entries at the P1 layer (entry)
  - Time trace entries at the P1 layer (exit)

A time trace entry shows a relative time and a time delta, both in seconds and milliseconds. A time trace entry with a time delta of more than one second is marked by an arrow at the right end of the trace entry line.

PDU Segment Trace Format

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

Each PDU segment data trace entry displays up to 16 data bytes, and consists of the following areas:

- The trace entry identifier (for example, EKAP1I_trc)
- The data displacement in hexadecimal format (for example, 00001A)
- The PDU data in hexadecimal character format, four groups of eight hexadecimal digits each
- The PDU data shown in character format, enclosed within asterisks

An example of an inbound probe and the corresponding outbound probe in a routing process trace with trace level 2 is shown below.

```
EKAP1I_trc PDU envelope data begin -------------------------------------------*
EKACSC_pxc 000000  004D0100  001C1001  0008A100  C3F4F1F0  * .(........˜.C410 *
EKACSC_pxc 000010  0009A101  E2C4C6C3  F10007A1  02C1F4C1  * ..˜.SDFC1..˜.A4A *
EKACSC_pxc 000020  00141101  0009A101  C8E4E2E7  F10007A1  * ......˜.HUSX1..˜ *
EKACSC_pxc 000030  02C1F1C1  00141403  0008A201  E7F1F4C1  * .A1A......s.X14A *
EKACSC_pxc 000040  0008A202  E7F4F1C1  0005B004  E3        * ..s.X41A....T    *
EKAP1I_trc PDU envelope data end ---------------------------------------------*


EKAP1R_trc PDU envelope data begin -------------------------------------------*
EKACSC_pxc 000000  00710100  001C1001  0008A100  C3F4F1F0  * .I........˜.C410 *
EKACSC_pxc 000010  0009A101  E2C4C6C3  F10007A1  02C1F4C1  * ..˜.SDFC1..˜.A4A *
EKACSC_pxc 000020  00141101  0009A101  C8E4E2E7  F10007A1  * ......˜.HUSX1..˜ *
EKACSC_pxc 000030  02C1F1C1  00141403  0008A201  E7F1F4C1  * .A1A......s.X14A *
EKACSC_pxc 000040  0008A202  E7F4F1C1  0005B004  E3002410  * ..s.X41A....T... *
EKACSC_pxc 000050  030007A1  0888A4A2  000CA109  4354DEEA  * ...˜.hus..˜.dhz2 *
EKACSC_pxc 000060  A6E95A5B  0008A10A  363490DF  0005B005  * wZ!$..˜...0∇....  *
EKACSC_pxc 000070  F0                                      * 0                *
EKAP1R_trc PDU envelope data end ---------------------------------------------*
```

Figure 12. Routing Process PDU Trace Example

Trace Facility Commands

The MERVA Link USS application control command application (ACC) provides commands for starting and stopping the MERVA Link processing trace facility, and for setting parameters such as the directory path to which traces are written, the trace level, and the wrap limit. These commands are described in the *MERVA for ESA Operations Guide*.

Intersystem Error Reporting

MERVA Link message transfer activities are performed by:

- Sending processes (also called client processes), which connect to a partner system and send messages

- Receiving processes (also called server processes), which accept an inbound connect request and receive messages
- Routing processes, which route conversations from a client to a server system if the client and server processes are not directly connected

An error can occur in any MERVA Link process. A failing process provides error information to local users and administrators as specified by the local MERVA Link installation. However, the MERVA Link system that hosts a failing process might be a remote system. To take appropriate action for error correction, information about the error must be made available to MERVA Link users and administrators at the local system.

This section describes the facilities provided by MERVA Link implementations in different operating system environments to report a routing or server process error to a local MERVA Link user or administrator.

Standard Error Information

Standard information about the status of a MERVA Link process consists of a status code (two numeric characters) and a diagnostic code (six alphanumeric characters). A status code of 08 or greater indicates a process error in all MERVA Link implementations. The diagnostic code provides information about the error in this case.

Status Code

Status codes 08 and 12 indicate a MERVA Link process error to a client MERVA Link user or administrator. These status code values are also used by a routing or server process to report a MERVA Link process error to the client system. All MERVA Link implementations (except MERVA Link OS/2) add one to the status code when they present error information of a MERVA Link routing or server process to a client MERVA Link user. Status codes 09 and 13 indicate a MERVA Link process error in a MERVA Link partner system. The diagnostic code associated with one of the latter status codes originates from a partner MERVA Link system.

Diagnostic Code

In case of a MERVA Link process error, the diagnostic code contains information about the specific error. The format and the meaning of this information is specified individually by each MERVA Link implementation. A small set of diagnostic codes that consist of a mnemonic is shared by some MERVA Link implementations. In general, it is necessary to know the type of the originating MERVA Link implementation, and to consult its messages and codes manual to understand the meaning of a diagnostic code.

Extended Error Information

Additional error information is defined by the MERVA Link architecture and supported in individual subsets by various MERVA Link implementations.

Diagnostic Code Originator Type

The reliable interpretation of a diagnostic code requires the knowlege of the type of the MERVA Link system that generated that diagnostic code. This is why a diagnostic code may be associated with an indicator of the type of the originating MERVA Link system, for example, MERVA Link ESA, MERVA Link USS, or MERVA Link AIX.

Internal Error Code Vector

MERVA Link processes use internally a set of return codes, reason codes, and other error codes for internal error reporting. These codes are the basis for generating a diagnostic code that reports an error to a process external resource. However, a subset of the available error information may be lost when the diagnostic code is generated. This is why some MERVA Link implementations provide a MERVA Link administrator with a set of internal codes called an *error code vector* (ECV). The ECV can be used for reference and detailed error analysis. Advanced knowledge of MERVA Link is required to interpret these error codes.

Error Explanation

The diagnostic code or the error code vector is the base information for error analysis and explanation of a MERVA Link process error. In most cases, a MERVA Link user or administrator must consult the appropriate MERVA Link documentation to perform that task.

To facilitate error analysis and the interpretation of MERVA Link process error information, some MERVA Link implementations provide error explanations in the form of a set of short operator message text lines. The currently available MERVA Link user interfaces do not support the display of error explanations. Error explanations are, however, accessible to a MERVA Link system administrator in MERVA Link traces and storage dumps.

Routing and Receiving Processes

MERVA Link routing and receiving processes that find an error use the applicable technique to indicate a conversation error, and send a MERVA Link error report to their partner process. Any error report contains the status code 08 or 12, and a diagnostic code of 6 alphanumeric characters.

Additional error information is included in an error report by some MERVA Link implementations. The related functions are described in the following separately for the various MERVA Link implementations. MERVA Link implementations that are not mentioned in the following provide only the minimum error report.

MERVA Link ESA Version 4

All MERVA Link receiving process implementations of MERVA ESA Version 4 (CICS ESA, CICS VSE, and IMS) support extended error reporting. The extended error reporting functions are, however, activated only if the partner system indicated its ability to accept an extended error report. The latter restriction provides compatibility with backlevel client MERVA Link implementations.

Diagnostic Code Originator Type: The diagnostic code originator type is indicated by the character **E** for MERVA Link **E**SA in the corresponding data element of an error report. All three MERVA Link ESA implementations share the same set of diagnostic and error codes. This is why the three MERVA Link ESA implementations are identified by the same diagnostic code originator type value.

MERVA Link ESA Internal Error Code Vector: The receiving process error code vector is included in the corresponding data element of an error report. It consists of six error codes (EC1 to EC6). The error code vector contains thefollowing error codes:

- When the error was found by an inbound TP (EKATR10 or EKATPI1), the error code vector contains the four return, reason, and error codes at the TP level as EC1 to EC4. EC5 and EC6 are zero.

- When the error was found by the inbound MTSP (EKASP10), the error code vector contains EC1 = 0008 and the four return, reason, and error codes at the MTSP level as EC2 to EC5. EC6 is zero.
- When the error was found by the inbound AS Program (EKAAR10), the error code vector contains EC1 = 0008, EC2 = 0008, and the four return, reason, and error codes at the AS level as EC3 to EC6.

MERVA Link USS

The MERVA Link USS receiving process implementation supports extended error reporting. This applies also to a routing process that is, primarily, a receiving process that performs also sending process functions. Except the diagnostic code originator type, the extended error reporting functions are, however, activated only if the partner system indicated its ability to accept an extended error report. The latter restriction provides compatibility with backlevel client MERVA Link implementations.

A MERVA Link USS routing process does not modify an error report received from a partner system if the client (sending) system did not indicate its ability to accept an extended error report.

Diagnostic Code Originator Type: The Diagnostic Code Originator Type **U** is part of any error report generated by a MERVA Link USS function. A minimum error report with this additional data element is acceptable by all MERVA Link implementations that are currently in service, and by all future MERVA Link implementations.

Error Explanation Text: A MERVA Link receiving or routing process calls its error explanation functions only if the partner system indicated its ability to accept an extended error report.

- Any error that originates from a MERVA Link USS function is reported by the standard means (status code, diagnostic code, and diagnostic code originator type). In addition, the MERVA Link USS diagnostic code is explained in a set of operator message lines that are included in the error report.
- An error report that originates from a partner MERVA Link ESA system may contain the applicable originator type data element and the internal error code vector data element. Both error information items are saved in the ACT ISC entry that describes the originator of the error report. ACC commands provide for displaying and explaining the error code vector.

  A MERVA Link USS routing process that finds this information in an error report and no explanation of the error, tries to explain the MERVA Link ESA error. It adds the set of operator messages to the error report that is returned by the MERVA Link USS function **ekaxev** (explain MERVA Link ESA error code vector). The first message of this set tells that the explanation of the error code vector has been performed by MERVA Link USS.

Sending Processes

MERVA Link sending processes are informed by their partner processes about an error. A sending process enters the receive state, and tries to receive a MERVA Link error report from its partner process. The data received in that situation may actually be an error report. It can, however, be any other unidentified data.

Unidentified error data is shown, for example, in traces or dumps, but not further analyzed. An error report received from a partner process is handled as specified by each MERVA Link implementation. The related functions are described in the following separately for the various MERVA Link implementations.

MERVA Link ESA Version 4

All MERVA Link sending process implementations of MERVA ESA Version 4 (CICS ESA, CICS VSE, and IMS) are able to receive an extended error report of about 1000 bytes from a partner system. The error report can be found in a MERVA Link sending transaction dump. It can also be found in a MERVA Link CICS full conversation trace.

MERVA Link USS

The MERVA Link USS sending process and the sending functions of a MERVA Link USS routing process support extended error reporting. An error report received from a partner system is scanned for extended error information. Diagnostic code originator type information and a MERVA Link ESA error code vector are saved in the applicable ACT ISC entry. ACC commands are provided to display and explain a MERVA Link ESA error code vector.

The following information is written to the MERVA Link USS process trace (if applicable):

- Operator message identifying the DC originator type
- Error report PDU in character and hexadecimal dump format
- One of the following, as far as available from the error report:
  1. Error explanation messages found in the error report
  2. MERVA Link ESA error code vector and its explanation
  3. Explanation of a diagnostic code originating from MERVA Link USS
  4. Explanation of a diagnostic code originating from MERVA Link AIX or MERVA Link NT
  5. Status code and diagnostic code values uninterpreted

The correctness of MERVA Link AIX and MERVA Link NT diagnostic code explanations by MERVA Link USS is not guaranteed. A corresponding disclaimer is included in the trace.

Error Report Log Facility

The MERVA Link USS Error Report Log Facility (ERR) provides for logging error information received from a partner system and error reports sent to a partner system in permanent error report log files. The error report log files are written to the **err** subdirectory of the applicable MERVA instance directory. There are separate error report log files for inbound and outbound error reports and for each partner node.

Error Report Log File

An error report log file is created if it is not already available when an error report log entry must be written. A failure to open the file is reported in the MERVA Link USS processing trace.

Error report log entries are always appended to the existing entries in an error log file. MERVA Link USS does never delete entries from an error report log file. A system administrator may wish to delete entries or remove complete files as appropriate.

Inbound Error Report Log File

The outbound MERVA Link USS TPs EKATPO and EKATCO append an error log entry to the applicable error report log file when error information is received from a partner system. The name of the error log file is the name of the partner node prefixed by **ier** (for inbound error report). The full path name of the inbound error report log file for partner node PNODE1 reads, for example, **/u/merva1/err/ier.PNODE1**.

Inbound Error Log Entry

An inbound error log entry starts with a heading that contains the time when the error information was received from the partner system. It is followed by the error information, an error report PDU, or unidentified error data.

The data in an error log entry is the same information that is written to the processing trace file by the MERVA Link USS internal function **ekaper()**. If an error report PDU is received, it is the PDU data in hexadecimal and character dump format, and one of the following:

- Error explanation lines, if operator message data elements are contained in the error report PDU
- An explanation of MERVA Link ESA receiving process error codes, if a MERVA Link ESA error code vector is contained in the error report PDU
- An explanation of the diagnostic code contained in the error report PDU, if the originator of this diagnostic code is MERVA Link ESA, AIX, or NT

The begin of unidentified error data is shown in three formats, hexadecimal, EBCDIC characters, and ASCII characters.

Outbound Error Report Log File

The inbound MERVA Link USS TPs EKATPI and EKATCI append an error log entry to the applicable error report log file when an error report is sent to a partner system. The name of the error log file is the name of the partner node prefixed by **oer** (for outbound error report). The full path name of the outbound error report log file for partner node PNODE1 reads, for example, **/u/merva1/err/oer.PNODE1**.

Outbound Error Log Entry

An outbound error log entry starts with a heading that contains the time when the error report was about to be sent to the partner system. The error report PDU follows in hexadecimal and character dump format.

ACC Commands That Handle Extended Error Information

The following application control command application (ACC) commands handle extended error information:

Analyze Routing Process to P-Node
The **arc** command tells whether a MERVA Link ESA error code vector (ECV) is available from the specified ACT ISC entry and, if it is, tells you which ACC command you need to enter to display an explanation of the ECV.

Display Status of ACT ISC Entry for P-Node
The command **dsc** tells whether a MERVA Link ESA error code vector (ECV) is available from the specified ACT ISC entry. If it is available, the dsc command output contains the ACC command that must be entered to display an explanation of the ECV.

Explain ECV contained in an ISC entry
The command **dxc** ***partner_node*** explains the error code vector (ECV) of a receiving process in a partner MERVA Link ESA node that is be contained in a MERVA Link USS ACT ISC entry. The argument ***partner_node*** is the name of the partner node that identifies the ISC entry.

Explain ECV for receiving process error of type AS
: The command **dxra** *as_ecv* displays an explanation of the specified error code vector (ECV) of a MERVA Link ESA receiving process error of type AS.

  If none of the error codes in the ECV are preceded by 00, you need not specify those leading zeros. However, if any of the error codes in the ECV are, you must specify each error code as 4 hexadecimal digits (including the leading zeros).

Explain ECV for receiving process error of type MT
: The command **dxrm** *mt_ecv* displays an explanation of the specified error code vector (ECV) of a MERVA Link ESA receiving process error of type MT.

  If none of the error codes in the ECV are preceded by 00, you need not specify those leading zeros. However, if any of the error codes in the ECV are, you must specify each error code as 4 hexadecimal digits (including the leading zeros).

Explain ECV for sending process error of type MT
: The command **dxsm** *mt_ecv* displays an explanation of the specified error code vector (ECV) of a MERVA Link ESA sending process error of type MT.

  If none of the error codes in the ECV are preceded by 00, you need not specify those leading zeros. However, if any of the error codes in the ECV are, you must specify each error code as 4 hexadecimal digits (including the leading zeros).

Part 5. Appendixes

Appendix A. PDU Data Elements

Level-1 Data Elements

The MERVA Link conversation trace contains conversation trace records. A conversation trace record consists of the conversation trace record prefix, the conversation trace control data element, and level 1 PDU data elements as conversation trace data.

A conversation trace record is a local matter. It is not transmitted to a partner system. Its data elements are defined as follows:

ID	Meaning of the Data Element
0010	Conversation Trace Record
8110	Conversation Trace Control Data

The other PDU data elements at the first level are:

ID	Meaning of the Data Element
0100	Probe Envelope
0101	Delivery Report Envelope
0102	Application Message PDU Envelope
0111	Delivery Report Content
0112	Status Report ASPDU
0120	IM-ASPDU Heading
0202	Command PDU Envelope
0220	Command Request PDU Heading
0221	Command Response PDU Heading
8121	IM-ASPDU Body Part Header
8122	IM-ASPDU Body Part Data Segment (EBCDIC)
8123	IM-ASPDU Body Part Encrypted Data Segment (EBCDIC)
8126	IM-ASPDU Body Part Compressed Data Segment (EBCDIC)
8127	IM-ASPDU Body Part Encrypted Compressed Data Segment (EBCDIC)
8132	IM-ASPDU Body Part Data Segment (ASCII)
8133	IM-ASPDU Body Part Encrypted Data Segment (ASCII)
8136	IM-ASPDU Body Part Compressed Data Segment (ASCII)
8137	IM-ASPDU Body Part Encrypted Compressed Data Segment (ASCII)
81FF	Message PDU Trailer
8221	Command Response PDU Body
82FF	Command PDU Trailer

Contents of Implicit Data Elements

Probe Envelope

A probe envelope (ID=0100) can contain the following data elements:

ID	Meaning of the Data Element
1001	Originator Address
1101	Recipient Address
1403	MTP Trace
1003	Client Security Information
1004	Change Security Information
B004	Probe Function

AMPDU Envelope

An AMPDU envelope (ID=0102) can contain the following data elements:

ID	Meaning of the Data Element
1001	Originator Address
1101	Recipient Address
9200	ASL Content Identifier
9201	MPDU Identifier
9301	Submit Time in YYMMDDHHMMSS Format
1403	MTP Trace
B000	Encoded Information Type
B001	Request for Delivery Notification
B002	Priority
B003	Content Type

Delivery Report Envelope and Content

The support of a MERVA Link delivery report and the associated data elements (ID=0101 and 0111) have been dropped in MERVA ESA Version 4.

Command PDU Envelope

A command PDU envelope (ID=0202) can contain the following data elements:

ID	Meaning of the Data Element
1001	Originator Address
1101	Recipient Address
B000	Encoded Information Type

Originator or Recipient Address

An originator or recipient address (ID=1001 or 1101) can contain the following data elements:

ID	Meaning of the Data Element
A100	MERVA System Type and Version
A101	Message Transfer Node Name
A102	MERVA Link ASP Name

Security Information

The security information data elements in a probe (ID=1003 or 1004) can contain the following data elements:

ID	Meaning of the Data Element
A108	User ID
A109	Encrypted Password
A10A	Password Encryption Control Information
B005	Password Encryption Method Identifier

MTP Trace

An MTP trace (ID=1403) can contain the following data elements:

ID	Meaning of the Data Element
A201	External Sending MTP Name
A202	External Receiving MTP Name

IM-ASPDU Heading

An IM-ASPDU heading (ID=0120) can contain the following data elements:

ID	Meaning of the Data Element
1002	Originator Application Descriptor
1102	Recipient Application Descriptor
9202	Inter-Application Messaging Message Identifier
9203	Message Integrity Protocol Message Identifier
9204	Message Integrity Protocol Message Sequence Number
9600	Message Body Encryption Indicator
9601	Message Authentication Indicator
9602	MERVA ESA Line Format Identifier
9603	MERVA ESA Message Type
9604	Message Integrity Protocol Window Size
9605	Message Body Compression Indicator
9608	Buckslip
9609	Subject
9610	Application Request Data
9611	Application Response Data
9612	Application Acknowledgment Data
9613	Application MAC Data
9614	Application PAC Data
B000	Body Part Type
B001	Request for Receipt Notification
B002	Priority
C000	Message Integrity Protocol Reset Indicator
C001	Possible Duplicate Message (PDM) Indicator

An IM-ASPDU Heading (ID=0120) can also contain application-defined data elements at any level. These data elements are passed between cooperating applications but they are not processed by standard MERVA Link Application Support. These data elements can be used by customer-written application support filters to add private information to a message heading.

Application Descriptor

An application descriptor (ID=1002 or 1102) can contain the following data elements:

ID	Meaning of the Data Element
A001	ASP Free Form Name
A101	Message Transfer Node Name
A102	MERVA Link ASP Name

Status Report ASPDU

A status report ASPDU (ID=0112) can contain the following data elements:

ID	Meaning of the Data Element
1102	Reported Recipient Application Descriptor
9202	Inter-Application Messaging Message Identifier
9203	Message Integrity Protocol Message Identifier
9204	Message Integrity Protocol Message Sequence Number
9604	Message Integrity Protocol Window Size
9608	Buckslip
9609	Subject
9611	Application Response Data
9612	Application Acknowledgment Data
9613	Application MAC Data
9614	Application PAC Data
C000	Message Integrity Protocol Reset Indicator
1500	Report

Report Data Element

A report data element (ID=1500) can contain the following data elements:

ID	Meaning of the Data Element
9301	Date/Time in YYMMDDHHMMSS Format
9501	Return Code
9502	Diagnostic Code
9503	Report Data
9605	MERVA Link Internal Error Code Vector
9606	Operator Message
B006	Diagnostic Code Originator Type

Command Request PDU Heading

A command request PDU heading (ID=0220) can contain the following data elements:

ID	Meaning of the Data Element
1002	Originator Application Descriptor
1102	Recipient Application Descriptor
9603	MERVA ESA Message Type
9608	Buckslip containing the operator command
960A	Unique MSC Correlation Data
960B	Specific MSC Correlation Data
960C	Specific CMD Correlation Data
B000	Body Part Type

Command Response PDU Heading

A command response PDU heading (ID=0221) can contain the following data elements:

ID	Meaning of the Data Element
1002	Originator Application Descriptor
1102	Recipient Application Descriptor
1500	Command Response Report
9603	MERVA ESA Message Type
9608	Buckslip containing a short operator message
960A	Unique MSC Correlation Data
960B	Specific MSC Correlation Data
B000	Body Part Type

Command Response Report

A command response report (ID=1500) can contain the following data elements:

ID	Meaning of the Data Element
9301	Date/Time in YYMMDDHHMMSS Format
9501	Return Code
9504	Command Processing Diagnostic Code

Application Defined Data Elements

The range of application defined data element identifiers is X'FF00' to X'FFFE' for explicit data elements and X'7F00' to X'7FFF' for implicit data elements. MERVA Link will not use any data element identifier starting with X'7F' or X'FF' in future extensions of the MERVA Link data stream architecture.

ID	Usage of the Data Element
7Fxx	Application private implicit data element
FFxx	Application private explicit data element

Any PDU can contain the “do not care” data element at any place on any level higher than one.

ID	Meaning of the Data Element
FFFF	Do not take care of this data element

List of Implicit Data Elements

| Implicit data elements are defined at levels 1 and 2.

| Level-1 Implicit Data Elements

The following table lists all implicit level-1 PDU data elements defined by MERVA Link in ascending sequence of their data element identifier.

ID	Meaning of the Data Element
0010	Conversation Trace Record
0100	Probe Envelope
0101	Delivery Report Envelope
0102	Application Message PDU Envelope
0111	Delivery Report Content
0112	Status Report ASPDU
0120	IM-ASPDU Heading
0202	Command PDU Envelope
0220	Command Request PDU Heading
0221	Command Response PDU Heading

Level-2 Implicit Data Elements

The following table lists all implicit level-2 PDU data elements defined by MERVA Link in ascending sequence of their data element identifier.

ID	Meaning of the Data Element
1001	Originator Address
1002	Originator Application Descriptor
1003	Client Security Information
1004	Change Security Information
1101	Recipient Address
1102	Recipient Application Descriptor
1403	MTP Trace
1500	Receipt Report, Error Report, Command Processing Report

List of Explicit Data Elements

Explicit data elements are defined at levels 1, 2, and 3.

Level-1 Explicit Data Elements

The following table lists all explicit level-1 PDU data elements defined by MERVA Link in ascending sequence of their data element identifier.

ID	Meaning of the Data Element
8110	Conversation Trace Control Data
8121	IM-ASPDU Body Part Header
8122	IM-ASPDU Body Part Data Segment (EBCDIC)
8123	IM-ASPDU Body Part Encrypted Data Segment (EBCDIC)
8126	IM-ASPDU Body Part Compressed Data Segment (EBCDIC)
8127	IM-ASPDU Body Part Encrypted Compressed Data Segment (EBCDIC)
8132	IM-ASPDU Body Part Data Segment (ASCII)
8133	IM-ASPDU Body Part Encrypted Data Segment (ASCII)
8136	IM-ASPDU Body Part Compressed Data Segment (ASCII)
8137	IM-ASPDU Body Part Encrypted Compressed Data Segment (ASCII)
81FF	Message PDU Trailer
8220	Command Request PDU Body
8221	Command Response PDU Body
82FF	Command PDU Trailer

Level-2 and Level-3 Explicit Data Elements

The following table lists all explicit level-2 and level-3 PDU data elements defined by MERVA Link in ascending sequence of their data element identifier.

ID	Meaning of the Data Element
9200	ASL Content Identifier
9201	MPDU Identifier
9202	Inter-Application Messaging Message Identifier
9203	Message Integrity Protocol Message Identifier
9204	Message Integrity Protocol Message Sequence Number
9301	Date and Time in YYMMDDHHMMSS Format
9501	Delivery or Receipt Return Code
9502	Message Processing Diagnostic Code
9503	Report Data
9504	Command Processing Diagnostic Code
9505	MERVA Link Internal Error Code Vector
9506	Operator Message, Error Code Explanation
9600	Message Body Encryption Indicator
9601	Message Authentication Indicator
9602	MERVA ESA Line Format Identifier
9603	MERVA ESA Message Type
9604	Message Integrity Protocol Window Size
9605	Message Body Compression Indicator
9608	Buckslip
9609	Subject
960A	Unique MSC Correlation Data
960B	Specific MSC Correlation Data
960C	Specific CMD Correlation Data
9610	Application Request Data
9611	Application Response Data
9612	Application Acknowledgment Data
9613	Application MAC Data
9614	Application PAC Data
A001	ASP Free Form Name
A100	MERVA System Type and Version
A101	Message Transfer Node Name
A102	MERVA Link ASP Name
A108	Security Information User ID
A109	Security Information Encrypted Password
A10A	Security Information Password Encryption Control Info
A201	External Sending MTP Name
A202	External Receiving MTP Name
B000	Encoded Information Type, Body Part Type
B001	Request for Delivery or Receipt Notification
B002	Priority
B003	Content Type
B004	Probe Function
B005	Password Encryption Method
B006	Diagnostic Code Originator Type
C000	Message Integrity Protocol Reset Indicator
C001	Possible Duplicate Message (PDM) Indicator

Appendix B. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Deutschland
Informationssysteme GmbH
Department 3982
Pascalstrasse 100

Trademarks

- MVS/ESA
- MVS/XA
- OS/2
- OS/390
- RACF
- VSE/ESA
- VTAM

Workstation (AWS) and Directory Services Application (DSA) are trademarks of S.W.I.F.T., La Hulpe in Belgium.

Pentium is a trademark of Intel Corporation.

PC Direct is a trademark of Ziff Communications Company in the United States, other countries, or both, and is used by IBM Corporation under license.

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

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

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

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

Glossary of Terms and Abbreviations

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

A

ACB. Access method control block.

ACC. MERVA Link USS application control command application. It provides a means of operating MERVA Link USS in USS shell and MVS batch environments.

Access method control block (ACB). A control block that links an application program to VSAM or VTAM.

ACD. MERVA Link USS application control daemon.

ACT. MERVA Link USS application control table.

address. See *SWIFT address*.

address expansion. The process by which the full name of a financial institution is obtained using the SWIFT address, telex correspondent's address, or a nickname.

AMPDU. Application message protocol data unit, which is defined in the MERVA Link P1 protocol, and consists of an envelope and its content.

answerback. In telex, the response from the dialed correspondent to the WHO R U signal.

answerback code. A group of up to 6 letters following or contained in the answerback. It is used to check the answerback.

APC. Application control.

API. Application programming interface.

APPC. Advanced Program-to-Program Communication based on SNA LU 6.2 protocols.

APPL. A VTAM definition statement used to define a VTAM application program.

application programming interface (API). An interface that programs can use to exchange data.

application support filter (ASF). In MERVA Link, a user-written program that can control and modify any data exchanged between the Application Support Layer and the Message Transfer Layer.

application support process (ASP). An executing instance of an application support program. Each application support process is associated with an ASP entry in the partner table. An ASP that handles outgoing messages is a *sending ASP*; one that handles incoming messages is a *receiving ASP*.

application support program (ASP). In MERVA Link, a program that exchanges messages and reports with a specific remote partener ASP. These two programs must agree on which conversation protocol they are to use.

ASCII. American Standard Code for Information Interchange. The standard code, using a coded set consisting of 7-bit coded characters (8 bits including parity check), used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

ASF. Application support filter.

ASF. (1) Application support process. (2) Application support program.

ASPDU. Application support protocol data unit, which is defined in the MERVA Link P2 protocol.

authentication. The SWIFT security check used to ensure that a message has not changed during transmission, and that it was sent by an authorized sender.

authenticator key. A set of alphanumeric characters used for the authentication of a message sent via the SWIFT network.

authenticator-key file. The file that stores the keys used during the authentication of a message. The file contains a record for each of your financial institution's correspondents.

B

Back-to-Back (BTB). A MERVA Link function that enables ASPs to exchange messages in the local MERVA Link node without using data communication services.

bank identifier code. A 12-character code used to identify a bank within the SWIFT network. Also called a SWIFT address. The code consists of the following subcodes:
- The bank code (4 characters)
- The ISO country code (2 characters)
- The location code (2 characters)
- The address extension (1 character)

- The branch code (3 characters) for a SWIFT user institution, or the letters "BIC" for institutions that are not SWIFT users.

Basic Security Manager (BSM). A component of VSE/ESA Version 2.4 that is invoked by the System Authorization Facility, and used to ensure signon and transaction security.

BIC. Bank identifier code.

BIC Bankfile. A tape of bank identifier codes supplied by S.W.I.F.T.

BIC Database Plus Tape. A tape of financial institutions and currency codes, supplied by S.W.I.F.T. The information is compiled from various sources and includes national, international, and cross-border identifiers.

BIC Directory Update Tape. A tape of bank identifier codes and currency codes, supplied by S.W.I.F.T., with extended information as published in the printed BIC Directory.

body. The second part of an IM-ASPDU. It contains the actual application data or the message text that the IM-AMPDU transfers.

BSC. Binary synchronous control.

BSM. Basic Security Manager.

BTB. Back-to-back.

buffer. A storage area used by MERVA programs to store a message in its internal format. A buffer has an 8-byte prefix that indicates its length.

C

CBT. SWIFT computer-based terminal.

CCSID. Coded character set identifier.

CDS. Control data set.

central service. In MERVA, a service that uses resources that either require serialization of access, or are only available in the MERVA nucleus.

CF message. Confirmed message. When a sending MERVA Link system is informed of the successful delivery of a message to the receiving application, it routes the delivered application messages as CF messages, that is, messages of class CF, to an ACK wait queue or to a complete message queue.

COA. Confirm on arrival.

COD. Confirm on delivery.

coded character set identifier (CCSID). The name of a coded set of characters and their code point assignments.

commit. In MQSeries, to commit operations is to make the changes on MQSeries queues permanent. After putting one or more messages to a queue, a commit makes them visible to other programs. After getting one or more messages from a queue, a commit permanently deletes them from the queue.

confirm-on-arrival (COA) report. An MQSeries report message type created when a message is placed on that queue. It is created by the queue manager that owns the destination queue.

confirm-on-delivery (COD) report. An MQSeries report message type created when an application retrieves a message from the queue in a way that causes the message to be deleted from the queue. It is created by the queue manager.

control fields. In MERVA Link, fields that are part of a MERVA message on the queue data set and of the message in the TOF. Control fields are written to the TOF at nesting identifier 0. Messages in SWIFT format do not contain control fields.

correspondent. An institution to which your institution sends and from which it receives messages.

correspondent identifier. The 11-character identifier of the receiver of a telex message. Used as a key to retrieve information from the Telex correspondents file.

cross-system coupling facility. See *XCF*.

coupling services. In a sysplex, the functions of XCF that transfer data and status information among the members of a group that reside in one or more of the MVS systems in the sysplex.

couple data set. See *XCF couple data set*.

CTP. MERVA Link command transfer processor.

currency code file. A file containing the currency codes, together with the name, fraction length, country code, and country names.

D

daemon. A long-lived process that runs unattended to perform continuous or periodic systemwide functions.

DASD. Direct access storage device.

data area. An area of a predefined length and format on a panel in which data can be entered or displayed. A field can consist of one or more data areas.

data element. A unit of data that, in a certain context, is considered indivisible. In MERVA Link, a data element consists of a 2-byte data element length field, a 2-byte data-element identifier field, and a field of variable length containing the data element data.

datagram. In TCP/IP, the basic unit of information passed across the Internet environment. This type of message does not require a reply, and is the simplest type of message that MQSeries supports.

data terminal equipment. That part of a data station that serves as a data source, data link, or both, and provides for the data communication control function according to protocols.

DB2. A family of IBM licensed programs for relational database management.

dead-letter queue. A queue to which a queue manager or application sends messages that it cannot deliver. Also called *undelivered-message queue*.

dial-up number. A series of digits required to establish a connection with a remote correspondent via the public telex network.

direct service. In MERVA, a service that uses resources that are always available and that can be used by several requesters at the same time.

display mode. The mode (PROMPT or NOPROMPT) in which SWIFT messages are displayed. See *PROMPT mode* and *NOPROMPT mode.*

distributed queue management (DQM). In MQSeries message queuing, the setup and control of message channels to queue managers on other systems.

DQM. Distributed queue management.

DTE. Data terminal equipment.

E

EBCDIC. Extended Binary Coded Decimal Interchange Code. A coded character set consisting of 8-bit coded characters.

ECB. Event control block.

EDIFACT. Electronic Data Interchange for Administration, Commerce and Transport (a United Nations standard).

ESM. External security manager.

EUD. End-user driver.

exception report. An MQSeries report message type that is created by a message channel agent when a message is sent to another queue manager, but that message cannot be delivered to the specified destination queue.

external line format (ELF) messages. Messages that are not fully tokenized, but are stored in a single field in the TOF. Storing messages in ELF improves performance, because no mapping is needed, and checking is not performed.

external security manager (ESM). A security product that is invoked by the System Authorization Facility. RACF is an example of an ESM.

F

FDT. Field definition table.

field. In MERVA, a portion of a message used to enter or display a particular type of data in a predefined format. A field is located by its position in a message and by its tag. A field is made up of one or more data areas. See also *data area*.

field definition table (FDT). The field definition table describes the characteristics of a field; for example, its length and number of its data areas, and whether it is mandatory. If the characteristics of a field change depending on its use in a particular message, the definition of the field in the FDT can be overridden by the MCB specifications.

field group. One or several fields that are defined as being a group. Because a field can occur more than once in a message, field groups are used to distinguish them. A name can be assigned to the field group during message definition.

field group number. In the TOF, a number is assigned to each field group in a message in ascending order from 1 to 255. A particular field group can be accessed using its field group number.

field tag. A character string used by MERVA to identify a field in a network buffer. For example, for SWIFT field 30, the field tag is **:30:**.

FIN. Financial application.

FIN-Copy. The MERVA component used for SWIFT FIN-Copy support.

finite state machine. The theoretical base describing the rules of a service request’s state and the conditions to state transitions.

FMT/ESA. MERVA-to-MERVA Financial Message Transfer/ESA.

form. A partially-filled message containing data that can be copied for a new message of the same message type.

G

GPA. General purpose application.

H

HFS. Hierarchical file system.

hierarchical file system (HFS). A system for organizing files in a hierarchy, as in a UNIX system. OS/390 UNIX System Services files are organized in an HFS. All files are members of a directory, and each directory is in turn a member of a directory at a higher level in the HFS. The highest level in the hierarchy is the root directory.

I

IAM. Interapplication messaging (a MERVA Link message exchange protocol).

IM-ASPDU. Interapplication messaging application support protocol data unit. It contains an application message and consists of a heading and a body.

incore request queue. Another name for the request queue to emphasize that the request queue is held in memory instead of on a DASD.

InetD. Internet Daemon. It provides TCP/IP communication services in the OS/390 USS environment.

initiation queue. In MQSeries, a local queue on which the queue manager puts trigger messages.

input message. A message that is input into the SWIFT network. An input message has an input header.

INTERCOPE TelexBox. This telex box supports various national conventions for telex procedures and protocols.

interservice communication. In MERVA ESA, a facility that enables communication among services if MERVA ESA is running in a multisystem environment.

intertask communication. A facility that enables application programs to communicate with the MERVA nucleus and so request a central service.

IP. Internet Protocol.

IP message. In-process message. A message that is in the process of being transferred to another application.

ISC. Intersystem communication.

ISN. Input sequence number.

ISN acknowledgment. A collective term for the various kinds of acknowledgments sent by the SWIFT network.

ISO. International Organization for Standardization.

ITC. Intertask communication.

J

JCL. Job control language.

journal. A chronological list of records detailing MERVA actions.

journal key. A key used to identify a record in the journal.

journal service. A MERVA central service that maintains the journal.

K

KB. Kilobyte (1024 bytes).

key. A character or set of characters used to identify an item or group of items. For example, the user ID is the key to identify a user file record.

key-sequenced data set (KSDS). A VSAM data set whose records are loaded in key sequence and controlled by an index.

keyword parameter. A parameter that consists of a keyword, followed by one or more values.

KSDS. Key-sequenced data set.

L

LAK. Login acknowledgment message. This message informs you that you have successfully logged in to the SWIFT network.

large message. A message that is stored in the large message cluster (LMC). The maximum length of a message to be stored in the VSAM QDS is 31900 bytes. Messages up to 2MB can be stored in the LMC. For queue management using DB2 no distinction is made between messages and large messages.

large queue element. A queue element that is larger than the smaller of:

- The limiting value specified during the customization of MERVA
- 32KB

LC message. Last confirmed control message. It contains the message-sequence number of the application or acknowledgment message that was last confirmed; that is, for which the sending MERVA Link system most recently received confirmation of a successful delivery.

LDS. Logical data stream.

LMC. Large message cluster.

LNK. Login negative acknowledgment message. This message indicates that the login to the SWIFT network has failed.

local queue. In MQSeries, a queue that belongs to a local queue manager. A local queue can contain a list of messages waiting to be processed. Contrast with *remote queue*.

local queue manager. In MQSeries, the queue manager to which the program is connected, and that provides message queuing services to that program. Queue managers to which a program is not connected are remote queue managers, even if they are running on the same system as the program.

login. To start the connection to the SWIFT network.

LR message. Last received control message, which contains the message-sequence number of the application or acknowledgment message that was last received from the partner application.

LSN. Login sequence number.

LT. See *LTERM*.

LTC. Logical terminal control.

LTERM. Logical terminal. Logical terminal names have 4 characters in CICS and up to 8 characters in IMS.

LU. A VTAM logical unit.

M

maintain system history program (MSHP). A program used for automating and controlling various installation, tailoring, and service activities for a VSE system.

MCA. Message channel agent.

MCB. Message control block.

MERVA ESA. The IBM licensed program Message Entry and Routing with Interfaces to Various Applications for ESA.

MERVA Link. A MERVA component that can be used to interconnect several MERVA systems.

message. A string of fields in a predefined form used to provide or request information. See also *SWIFT financial message.*

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

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

message channel. In MQSeries distributed message queuing, a mechanism for moving messages from one queue manager to another. A message channel comprises two message channel agents (a sender and a receiver) and a communication link.

message channel agent (MCA). In MQSeries, a program that transmits prepared messages from a transmission queue to a communication link, or from a communication link to a destination queue.

message control block (MCB). The definition of a message, screen panel, net format, or printer layout made during customization of MERVA.

Message Format Service (MFS). A MERVA direct service that formats a message according to the medium to be used, and checks it for formal correctness.

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

Message Integrity Protocol (MIP). In MERVA Link, the protocol that controls the exchange of messages between partner ASPs. This protocol ensures that any loss of a message is detected and reported, and that no message is duplicated despite system failures at any point during the transfer process.

message-processing function. The various parts of MERVA used to handle a step in the message-processing route, together with any necessary equipment.

message queue. See *queue*.

Message Queue Interface (MQI). The programming interface provided by the MQSeries queue managers. It provides a set of calls that let application programs access message queuing services such as sending messages, receiving messages, and manipulating MQSeries objects.

Message Queue Manager (MQM). An IBM licensed program that provides message queuing services. It is part of the MQSeries set of products.

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

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

message type (MT). A number, up to 7 digits long, that identifies a message. SWIFT messages are identified by a 3-digit number; for example SWIFT message type MT S100.

MFS. Message Format Service.

MIP. Message Integrity Protocol.

MPDU. Message protocol data unit, which is defined in P1.

MPP. In IMS, message-processing program.

MQA. MQ Attachment.

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

MQH. MQSeries queue handler.

MQI. Message queue interface.

MQM. Message queue manager.

MQS. MQSeries nucleus server.

MQSeries. A family of IBM licensed programs that provides message queuing services.

MQSeries nucleus server (MQS). A MERVA component that listens for messages on an MQI queue, receives them, extracts a service request, and passes it via the request queue handler to another MERVA ESA instance for processing.

MQSeries queue handler (MQH). A MERVA component that performs service calls to the Message Queue Manager via the provided Message Queue Interface.

MRN. Message reference number.

MSC. MERVA system control facility.

MSHP. Maintain system history program.

MSN. Message sequence number.

MT. Message type.

MTP. (1) Message transfer program. (2) Message transfer process.

MTS. Message Transfer System.

MTSP. Message Transfer Service Processor.

MTT. Message type table.

multisystem application. (1) An application program that has various functions distributed across MVS systems in a multisystem environment. (2) In XCF, an authorized application that uses XCF coupling services. (3) In MERVA ESA, multiple instances of MERVA ESA that are distributed among different MVS systems in a multisystem environment.

multisystem environment. An environment in which two or more MVS systems reside on one or more processors, and programs on one system can communicate with programs on the other systems. With XCF, the environment in which XCF services are available in a defined sysplex.

multisystem sysplex. A sysplex in which one or more MVS systems can be initialized as part of the sysplex. In a multisystem sysplex, XCF provides coupling services on all systems in the sysplex and requires an XCF couple data set that is shared by all systems. See also *single-system sysplex*.

MVS/ESA. Multiple Virtual Storage/Enterprise Systems Architecture.

N

namelist. An MQSeries for MVS/ESA object that contains a list of queue names.

nested message. A message that is composed of one or more message types.

nested message type. A message type that is contained in another message type. In some cases, only part of a message type (for example, only the mandatory fields) is nested, but this “partial” nested message type is also considered to be nested. For example, SWIFT MT 195 could be used to request information about a SWIFT MT 100 (customer transfer). The SWIFT MT 100 (or at least its mandatory fields) is then nested in SWIFT MT 195.

nesting identifier. An identifier (a number from 2 to 255) that is used to access a nested message type.

network identifier. A single character that is placed before a message type to indicate which network is to be used to send the message; for example, **S** for SWIFT

network service access point (NSAP). The endpoint of a network connection used by the SWIFT transport layer.

NOPROMPT mode. One of two ways to display a message panel. NOPROMPT mode is only intended for experienced SWIFT Link users who are familiar with the structure of SWIFT messages. With NOPROMPT mode, only the SWIFT header, trailer, and pre-filled fields and their tags are displayed. Contrast with *PROMPT mode*.

NSAP. Network service access point.

nucleus server. A MERVA component that processes a service request as selected by the request queue handler. The service a nucleus server provides and the way it provides it is defined in the nucleus server table (DSLNSVT).

O

object. In MQSeries, objects define the properties of queue managers, queues, process definitions, and namelists.

occurrence. See *repeatable sequence*.

option. One or more characters added to a SWIFT field number to distinguish among different layouts for and meanings of the same field. For example, SWIFT field 60 can have an option F to identify a first opening balance, or M for an intermediate opening balance.

origin identifier (origin ID). A 34-byte field of the MERVA user file record. It indicates, in a MERVA and SWIFT Link installation that is shared by several banks, to which of these banks the user belongs. This lets the user work for that bank only.

OSN. Output sequence number.

OSN acknowledgment. A collective term for the various kinds of acknowledgments sent to the SWIFT network.

output message. A message that has been received from the SWIFT network. An output message has an output header.

P

P1. In MERVA Link, a peer-to-peer protocol used by cooperating message transfer processes (MTPs).

P2. In MERVA Link, a peer-to-peer protocol used by cooperating application support processes (ASPs).

P3. In MERVA Link, a peer-to-peer protocol used by cooperating command transfer processors (CTPs).

packet switched public data network (PSPDN). A public data network established and operated by network common carriers or telecommunication administrations for providing packet-switched data transmission.

panel. A formatted display on a display terminal. Each page of a message is displayed on a separate panel.

parallel processing. The simultaneous processing of units of work by several servers. The units of work can be either transactions or subdivisions of larger units of work.

parallel sysplex. A sysplex that uses one or more coupling facilities.

partner table (PT). In MERVA Link, the table that defines how messages are processed. It consists of a header and different entries, such as entries to specify the message-processing parameters of an ASP or MTP.

PCT. Program Control Table (of CICS).

PDE. Possible duplicate emission.

PDU. Protocol data unit.

PF key. Program-function key.

positional parameter. A parameter that must appear in a specified location relative to other parameters.

PREMIUM. The MERVA component used for SWIFT PREMIUM support.

process definition object. An MQSeries object that contains the definition of an MQSeries application. A queue manager uses the definitions contained in a process definition object when it works with trigger messages.

program-function key. A key on a display terminal keyboard to which a function (for example, a command) can be assigned. This lets you execute the function (enter the command) with a single keystroke.

PROMPT mode. One of two ways to display a message panel. PROMPT mode is intended for SWIFT Link users who are unfamiliar with the structure of SWIFT messages. With PROMPT mode, all the fields and tags are displayed for the SWIFT message. Contrast with *NOPROMPT mode*.

protocol data unit (PDU). In MERVA Link a PDU consists of a structured sequence of implicit and explicit data elements:
- Implicit data elements contain other data elements.
- Explicit data elements cannot contain any other data elements.

PSN. Public switched network.

PSPDN. Packet switched public data network.

PSTN. Public switched telephone network.

PT. Partner table.

PTT. A national post and telecommunication authority (post, telegraph, telephone).

Q

QDS. Queue data set.

QSN. Queue sequence number.

queue. (1) In MERVA, a logical subdivision of the MERVA queue data set used to store the messages associated with a MERVA message-processing function. A queue has the same name as the message-processing function with which it is associated. (2) In MQSeries, an object onto which message queuing applications can put messages, and from which they can get messages. A queue is owned and maintained by a queue manager. See also *request queue*.

queue element. A message and its related control information stored in a data record in the MERVA ESA Queue Data Set.

queue management. A MERVA service function that handles the storing of messages in, and the retrieval of messages from, the queues of message-processing functions.

queue manager. (1) An MQSeries system program that provides queueing services to applications. It provides an application programming interface so that programs can access messages on the queues that the queue manager owns. See also *local queue manager* and *remote queue manager*. (2) The MQSeries object that defines the attributes of a particular queue manager.

queue sequence number (QSN). A sequence number that is assigned to the messages stored in a logical queue by MERVA ESA queue management in ascending order. The QSN is always unique in a queue. It is reset to zero when the queue data set is formatted, or when a queue management restart is carried out and the queue is empty.

R

RACF. Resource Access Control Facility.

RBA. Relative byte address.

RC message. Recovered message; that is, an IP message that was copied from the control queue of an inoperable or closed ASP via the **recover** command.

ready queue. A MERVA queue used by SWIFT Link to collect SWIFT messages that are ready for sending to the SWIFT network.

remote queue. In MQSeries, a queue that belongs to a remote queue manager. Programs can put messages on remote queues, but they cannot get messages from remote queues. Contrast with *local queue*.

remote queue manager. In MQSeries, a queue manager is remote to a program if it is not the queue manager to which the program is connected.

repeatable sequence. A field or a group of fields that is contained more than once in a message. For example, if the SWIFT fields 20, 32, and 72 form a sequence, and if this sequence can be repeated up to 10 times in a message, each sequence of the fields 20, 32, and 72 would be an occurrence of the repeatable sequence.

In the TOF, the occurrences of a repeatable sequence are numbered in ascending order from 1 to 32767 and can be referred to using the occurrence number.

A repeatable sequence in a message may itself contain another repeatable sequence. To identify an occurrence within such a nested repeatable sequence, more than one occurrence number is necessary.

reply message. In MQSeries, a type of message used for replies to request messages.

reply-to queue. In MQSeries, the name of a queue to which the program that issued an MQPUT call wants a reply message or report message sent.

report message. In MQSeries, a type of message that gives information about another message. A report message usually indicates that the original message cannot be processed for some reason.

request message. In MQSeries, a type of message used for requesting a reply from another program.

request queue. The queue in which a service request is stored. It resides in main storage and consists of a set of request queue elements that are chained in different queues:

- Requests waiting to be processed
- Requests currently being processed
- Requests for which processing has finished

request queue handler (RQH). A MERVA ESA component that handles the queueing and scheduling of service requests. It controls the request processing of a nucleus server according to rules defined in the finite state machine.

Resource Access Control Facility (RACF). An IBM licensed program that provides for access control by identifying and verifying users to the system, authorizing access to protected resources, logging detected unauthorized attempts to enter the system, and logging detected accesses to protected resources.

retype verification. See *verification*.

routing. In MERVA, the passing of messages from one stage in a predefined processing path to the next stage.

RP. Regional processor.

RQH. Request queue handler.

RRDS. Relative record data set.

S

SAF. System Authorization Facility.

SCS. SNA character string

SCP. System control process.

SDI. Sequential data set input. A batch utility used to import messages from a sequential data set or a tape into MERVA ESA queues.

SDO. Sequential data set output. A batch utility used to export messages from a MERVA ESA queue to a sequential data set or a tape.

SDY. Sequential data set system printer. A batch utility used to print messages from a MERVA ESA queue.

service request. A type of request that is created and passed to the request queue handler whenever a nucleus server requires a service that is not currently available.

sequence number. A number assigned to each message exchanged between two nodes. The number is increased by one for each successive message. It starts from zero each time a new session is established.

sign off. To end a session with MERVA.

sign on. To start a session with MERVA.

single-system sysplex. A sysplex in which only one MVS system can be initialized as part of the sysplex. In a single-system sysplex, XCF provides XCF services on the system, but does not provide signalling services between MVS systems. A single-system sysplex requires an XCF couple data set. See also *multisystem sysplex*.

small queue element. A queue element that is smaller than the smaller of:

- The limiting value specified during the customization of MERVA
- 32KB

SMP/E. System Modification Program Extended.

SN. Session number.

SNA. Systems network architecture.

SNA character string. In SNA, a character string composed of EBCDIC controls, optionally mixed with user data, that is carried within a request or response unit.

SPA. Scratch pad area.

SQL. Structured Query Language.

SR-ASPDU. The status report application support PDU, which is used by MERVA Link for acknowledgment messages.

SSN. Select sequence number.

subfield. A subdivision of a field with a specific meaning. For example, the SWIFT field 32 has the subfields date, currency code, and amount. A field can have several subfield layouts depending on the way the field is used in a particular message.

SVC. (1) Switched Virtual Circuit. (2) Supervisor call instruction.

S.W.I.F.T. (1) Society for Worldwide Interbank Financial Telecommunication s.c. (2) The network provided and managed by the Society for Worldwide Interbank Financial Telecommunication s.c.

SWIFT address. Synonym for *bank identifier code*.

SWIFT Correspondents File. The file containing the bank identifier code (BIC), together with the name, postal address, and zip code of each financial institution in the BIC Directory.

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

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

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

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

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

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

SWIFT system message. A SWIFT general purpose application (GPA) message or a financial application (FIN) message in SWIFT category 0.

switched virtual circuit (SVC). An X.25 circuit that is dynamically established when needed. It is the X.25 equivalent of a switched line.

sysplex. One or more MVS systems that communicate and cooperate via special multisystem hardware components and software services.

System Authorization Facility (SAF). An MVS or VSE facility through which MERVA ESA communicates with an external security manager such as RACF (for MVS) or the basic security manager (for VSE).

System Control Process (SCP). A MERVA Link component that handles the transfer of MERVA ESA commands to a partner MERVA ESA system, and the receipt of the command response. It is associated with a system control process entry in the partner table.

System Modification Program Extended (SMP/E). A licensed program used to install software and software changes on MVS systems.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operating sequences for transmitting information units through, and for controlling the configuration and operation of, networks.

T

tag. A field identifier.

TCP/IP. Transmission Control Protocol/Internet Protocol.

Telex Correspondents File. A file that stores data about correspondents. When the user enters the corresponding nickname in a Telex message, the corresponding information in this file is automatically retrieved and entered into the Telex header area.

telex header area. The first part of the telex message. It contains control information for the telex network.

telex interface program (TXIP). A program that runs on a Telex front-end computer and provides a communication facility to connect MERVA ESA with the Telex network.

Telex Link. The MERVA ESA component used to link to the public telex network via a Telex substation.

Telex substation. A unit comprised of the following:

- Telex Interface Program
- A Telex front-end computer
- A Telex box

Terminal User Control Block (TUCB). A control block containing terminal-specific and user-specific information used for processing messages for display devices such as screen and printers.

test key. A key added to a telex message to ensure message integrity and authorized delivery. The test key is an integer value of up to 16 digits, calculated manually or by a test-key processing program using the significant information in the message, such as amounts, currency codes, and the message date.

test-key processing program. A program that automatically calculates and verifies a test key. The Telex Link supports panels for input of test-key-related data and an interface for a test-key processing program.

TFD. Terminal feature definitions table.

TID. Terminal identification. The first 9 characters of a bank identifier code (BIC).

TOF. Originally the abbreviation of *tokenized form*, the TOF is a storage area where messages are stored so that their fields can be accessed directly by their field names and other index information.

TP. Transaction program.

transaction. A specific set of input data that triggers the running of a specific process or job; for example, a message destined for an application program.

transaction code. In IMS and CICS, an alphanumeric code that calls an IMS message processing program or a CICS transaction. Transaction codes have 4 characters in CICS and up to 8 characters in IMS.

Transmission Control Protocol/Internet Protocol (TCP/IP). A set of communication protocols that support peer-to-peer connectivity functions for both local and wide area networks.

transmission queue. In MQSeries, a local queue on which prepared messages destined for a remote queue manager are temporarily stored.

trigger event. In MQSeries, an event (such as a message arriving on a queue) that causes a queue manager to create a trigger message on an initiation queue.

trigger message. In MQSeries, a message that contains information about the program that a trigger monitor is to start.

trigger monitor. In MQSeries, a continuously-running application that serves one or more initiation queues. When a trigger message arrives on an initiation queue, the trigger monitor retrieves the message. It uses the information in the trigger message to start a process that serves the queue on which a trigger event occurred.

triggering. In MQSeries, a facility that allows a queue manager to start an application automatically when predetermined conditions are satisfied.

TUCB. Terminal User Control Block.

TXIP. Telex interface program.

U

UMR. Unique message reference.

unique message reference (UMR). An optional feature of MERVA ESA that provides each message with a unique identifier the first time it is placed in a queue. It is composed of a MERVA ESA installation name, a sequence number, and a date and time stamp.

UNIT. A group of related literals or fields of an MCB definition, or both, enclosed by a DSLLUNIT and DSLLUEND macroinstruction.

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

- An application program interface (API)
- An interactive shell interface

UN/EDIFACT. United Nations Standard for Electronic Data Interchange for Administration, Commerce and Transport.

USE. S.W.I.F.T. User Security Enhancements.

user file. A file containing information about all MERVA ESA users; for example, which functions each user is allowed to access. The user file is encrypted and can only be accessed by authorized persons.

user identification and verification. The acts of identifying and verifying a RACF-defined user to the system during logon or batch job processing. RACF identifies the user by the user ID and verifies the user by the password or operator identification card supplied during logon processing or the password supplied on a batch JOB statement.

USS. UNIX System Services.

V

verification. Checking to ensure that the contents of a message are correct. Two kinds of verification are:

- Visual verification: you read the message and confirm that you have done so
- Retype verification: you reenter the data to be verified

Virtual LU. An LU defined in MERVA Extended Connectivity for communication between MERVA and MERVA Extended Connectivity.

Virtual Storage Access Method (VSAM). An access method for direct or sequential processing of fixed and variable-length records on direct access devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry sequence), or by relative-record number.

Virtual Telecommunications Access Method (VTAM). An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability.

VSAM. Virtual Storage Access Method.

VTAM. Virtual Telecommunications Access Method (IBM licensed program).

W

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

X

X.25. An ISO standard for interface to packet switched communications services.

XCF. Abbreviation for *cross-system coupling facility*, which is a special logical partition that provides high-speed caching, list processing, and locking functions in a sysplex. XCF provides the MVS coupling services that allow authorized programs on MVS systems in a multisystem environment to communicate with (send data to and receive data from) authorized programs on other MVS systems.

XCF couple data sets. A data set that is created through the XCF couple data set format utility and, depending on its designated type, is shared by some or all of the MVS systems in a sysplex. It is accessed only by XCF and contains XCF-related data about the sysplex, systems, applications, groups, and members.

XCF group. The set of related members defined to SCF by a multisystem application in which members of the group can communicate with (send data to and receive data from) other members of the same group. All MERVA systems working together in a sysplex must pertain to the same XCF group.

XCF member. A specific function of a multisystem application that is defined to XCF and assigned to a group by the multisystem application. A member resides on one system in a sysplex and can use XCF services to communicate with other members of the same group.

Bibliography

MERVA ESA Publications

- *MERVA for ESA Version 4: Application Programming Interface Guide*, SH12-6374
- *MERVA for ESA Version 4: Advanced MERVA Link*, SH12-6390
- *MERVA for ESA Version 4: Concepts and Components*, SH12-6381
- *MERVA for ESA Version 4: Customization Guide*, SH12-6380
- *MERVA for ESA Version 4: Diagnosis Guide*, SH12-6382
- *MERVA for ESA Version 4: Installation Guide*, SH12-6378
- *MERVA for ESA Version 4: Licensed Program Specifications*, GH12-6373
- *MERVA for ESA Version 4: Macro Reference*, SH12-6377
- *MERVA for ESA Version 4: Messages and Codes*, SH12-6379
- *MERVA for ESA Version 4: Operations Guide*, SH12-6375
- *MERVA for ESA Version 4: System Programming Guide*, SH12-6366
- *MERVA for ESA Version 4: User's Guide*, SH12-6376

MERVA ESA Components Publications

- *MERVA Automatic Message Import/Export Facility: User's Guide*, SH12-6389
- *MERVA Connection/NT*, SH12-6339
- *MERVA Connection/400*, SH12-6340
- *MERVA Directory Services*, SH12-6367
- *MERVA Extended Connectivity: Installation and User's Guide*, SH12-6157
- *MERVA Message Processing Client for Windows NT: User's Guide*, SH12-6341
- *MERVA Traffic Reconciliation*, SH12-6392
- *MERVA USE: Administration Guide*, SH12-6338
- *MERVA USE & Branch for Windows NT: User's Guide*, SH12-6334
- *MERVA USE & Branch for Windows NT: Installation and Customization Guide*, SH12-6335
- *MERVA USE & Branch for Windows NT: Application Programming Guide*, SH12-6336
- *MERVA USE & Branch for Windows NT: Diagnosis Guide*, SH12-6337
- *MERVA USE & Branch for Windows NT: Migration Guide*, SH12-6393
- *MERVA USE & Branch for Windows NT: Installation and Customization Guide*, SH12-6335
- *MERVA Workstation Based Functions*, SH12-6383

Other IBM Publications

- *CICS Transaction Server for OS/390: CICS Distributed Transaction Programming Guide*, SC33-1691
- *CICS Transaction Server for OS/390: CICS Intercommunication Guide*, SC33-1695
- *CICS Transaction Server for OS/390: CICS RACF Security Guide*, SC33-1701
- *OS/390 MVS Writing TPs for APPC/MVS*, GC28-1775
- *OS/390 MVS Planning: APPC/MVS Management*, GC28-1807
- *OS/390 UNIX System Services User's Guide*, SC28-1891
- *OS/390 UNIX System Services Command Reference*, SC28-1892
- *OS/390 Security Server (RACF) Introduction*, GC28-1912.
- *High Level Assembler Language Reference*, SC26-4940

S.W.I.F.T. Publications

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

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

Index

MERVA Requirement Request

Use the form overleaf to send us requirement requests for the MERVA product. Fill in the blank lines with the information that we need to evaluate and implement your request. Provide also information about your hardware and software environments and about the MERVA release levels used in your environment.

Provide a detailed description of your requirement. If you are requesting a new function, describe in full what you want that function to do. If you are requesting that a function be changed, briefly describe how the function works currently, followed by how you are requesting that it should work.

If you are a customer, provide us with the appropriate contacts in your organization to discuss the proposal and possible implementation alternatives.

If you are an IBM employee, include at least the name of one customer who has this requirement. Add the name and telephone number of the appropriate contacts in the customer's organization to discuss the proposal and possible implementation alternatives. If possible, send this requirement online to MERVAREQ at SDFVM1.

For comments on this book, use the form provided at the back of this publication.

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

Send the fax to:

```
To: MERVA Development, Dept. 5640              Fax Number: +49-7031-16-4881
    Attention: Gerhard Stubbe                  Internet address:
                                               mervareq@de.ibm.com
    IBM Deutschland Entwicklung GmbH
    Schoenaicher Str. 220
    D-71032 Boeblingen
    Germany
```

MERVA Requirement Request

```
To: MERVA Development, Dept. 5640                  Fax Number: +49-7031-16-4881
    Attention: Gerhard Strubbe                      Internet address:
                                                    mervareq@de.ibm.com
    IBM Deutschland Entwicklung GmbH
    Schoenaicher Str. 220
    D-71032 Boeblingen       Germany
```

Page 1 of ______

Customer's Name	______________________
Customer's Address	______________________
	______________________
	______________________
Customer's Telephone/Fax	______________________
Contact Person at Customer's Location	______________________
Telephone/Fax	______________________
MERVA Version/Release	______________________
Operating System	______________________
Sub-System Version/Release	______________________
Hardware	______________________
Requirement Description	______________________
	______________________
	______________________
	______________________
	______________________
	______________________
	______________________
Expected Benefits	______________________
	______________________
	______________________

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

MERVA for ESA
Advanced MERVA Link
Version 4 Release 1

Publication No. SH12-6390-01

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
SH12-6390-01

Cut or Fold
Along Line

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

PLACE
POSTAGE
STAMP
HERE

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

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

SH12-6390-01

Cut or Fold
Along Line

Program Number: 5648-B29

SH12-6390-01

Spine information:

IBM MERVA for ESA Advanced MERVA Link Version 4 Release 1