BEA eLink Adapter for Mainframe

User Guide

BEA eLink Adapter for Mainframe 4.0
Document Edition 4.0
January 2001

BEA® eLink™ Adapter for Mainframe User Guide

Document Edition	Part Number	Date	Software Version
4.0		January 2001	BEA eLink Adapter for Mainframe, 4.0

Contents

About This Document

This guide provides information about BEA eLink Adapter for Mainframe (eAM), an ATMI platform multi-domain connectivity product that enables client/server transactions between OS390/CICS or IMS programs and ATMI applications via a Systems Network Architecture (SNA) network.

This section covers the following topics:

- How this Guide Is Organized
- Who Should Read This Information
- Product Documentation
- Contact Us

How this Guide Is Organized

The *BEA eLink Adapter for Mainframe User Guide* is organized as follows:

- Understanding the BEA eLink Adapter for Mainframe Solution

  This section gives an overview of the eAM solution.

- Configuring the System

  This section provides instructions for configuring the eAM software based on the configuration of your system.

- Verifying the Software

  This section provides information to build verification tests and run sample applications after the eAM software is installed and configured.

- Security

  The section provides instructions to set up security enhancements of the eAM software, which includes encryption and TCP/IP link authentication.

- Data Translations

  This section provides information about the data translation options such as data conversion, translation rules, and data mapping.

- APPC/IMS Programming Considerations

  This section provides information about using APPC protocols to enter IMS transactions.

- Integrating eAM with Crossplex

  This section provides information about integrating eAM with CrossPlex software.

Who Should Read This Information

The target audience for this document is primarily ATMI platform system application administrators and operators.

Administrators

As the application administrator of an ATMI platform, you will configure eAM using the `DMCONFIG` file and its associated *dm* commands. You must have sufficient SNA knowledge to configure the underlying SNA *stack* so it conforms with definitions created in VTAM and CICS for each remote domain. This document provides information to help you understand the relationship between ATMI platform configuration settings and SNA-based application configuration concepts.

Successfully linking and establishing conversations between ATMI platform applications and SNA-based programs requires special coordination. The names and characteristics of SNA resources, configured in the SNA stack, must agree with resources and characteristics defined in VTAM and CICS. This guide includes examples of these relationships.

Typically, remote VTAM and CICS resources are defined by system personnel in a data center where IBM mainframes are located. Therefore, you need to request the remote names of eAM and CICS resources from data center systems personnel and use those names to configure the local SNA.

Operators

As an operator of an ATMI platform, you will use existing skills with the ATMI platform domain administration facility to modify SNA domain configurations and get information about the configuration and runtime environment.

Product Documentation

The eAM documentation consists of the following:

- *BEA eLink Adapter for Mainframe Release Notes*
- *BEA eLink Adapter for Mainframe Installation Guide*
- *BEA eLink Adapter for Mainframe User Guide*
- *BEA CRM Administration Guide*

Recommended Reading

You should read the appropriate ATMI platform product documentation, which is essential to comprehending the material in this document.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation or go directly to the “e-docs” Product Documentation page at `http://e-docs.beasys.com`.

How to Print the Document

You can print a copy of this document from a Web browser, one file at a time, by using the File—>Print option on your Web browser.

A PDF version of this document is available on the eAM documentation Home page on the e-docs Web site (and also on the documentation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the eAM documentation Home page, click the PDF files button, and select the document you want to print.

If you do not have the Adobe Acrobat Reader, you can get it for free from the Adobe Web site at `http://www.adobe.com/`.

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.

Convention	Item

`monospace text`	Indicates code samples, commands and their options, data structures and their members, data types, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 SIGNON OR
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.

`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f` *`file-list`*`]...`
	`[-l` *`file-list`*`]...`

`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.

Convention	Item
`...`	Indicates one of the following in a command line: ■ That an argument can be repeated several times in a command line ■ That the statement omits additional optional arguments ■ That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. *Example*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

Contact Us

Your feedback on the BEA eLink Adapter for Mainframe, 4.0 documentation is important to us. Send us e-mail at `docsupport@beasys.com` if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the eLink Adapter for Mainframe, 4.0 documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA eLink Adapter for Mainframe 4.0 release.

If you have any questions about this version of BEA eLink Adapter for Mainframe, 4.0, or if you have problems installing and running BEA eLink Adapter for Mainframe, 4.0, contact BEA Customer Support through BEA WebSupport at `www.beasys.com`. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes

- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

CHAPTER

1 Understanding the BEA eLink Adapter for Mainframe Solution

In today’s rapidly changing environment, businesses must continue to address changes in their business needs through adjustments in their corresponding computer infrastructure. Packaged applications successfully automate many internal operations such as financial, manufacturing, and human resources tasks. However, these applications’ data formats and interface protocols may be proprietary and much of the applications’ functionality may not be accessible from any exposed API. The consequence is isolation of these systems, loss of flexibility, and the inability to change at a later time.

When business systems need to share information and capabilities to operate efficiently or to expand to the web, integration problems surface. Businesses need a greater exchange of information with systems that communicate at both a database and a process level within the organization as well as with customers’ and suppliers’ systems. Businesses need to develop systems that are open, robust, and flexible to change while retaining the systems they have already purchased, developed, or inherited.

Enterprise Application Integration

Enterprise Application Integration (EAI) meets these business needs by providing a solution for integrating computer systems and applications. EAI requires that the applications exchange function and data messages seamlessly with minimal or no change to the applications themselves. Whether streamlining is accomplished through web-enablement, the consolidation of user interfaces, or the removal of redundant applications, the primary goal is always to simplify the system and reduce the effort required to maintain and operate it. An EAI system must:

- Be unaffected by system interfaces and environments
- Be highly reliable
- Be scalable
- Operate in real time or batch
- Support synchronous or asynchronous communications
- Support reliable queuing
- Support transactions
- Be manageable
- Support data integration capabilities
- Support business process flow control

BEA eLink Adapter for Mainframe Overview

BEA eLink Adapter for Mainframe (eAM) meets EAI requirements by allowing ATMI applications to communicate with the mainframe. The BEA eLink family of products provides domain-compliant gateways that permit administration of the remote Transaction Processing (TP) system as a foreign domain.

The eAM domain architecture extends the scope of ATMI platforms to provide coordinated transaction processing across an enterprise's geographic or organizational boundaries. Within each domain, the administrator determines which local services are available to other specific domains, thus enabling client applications to request those services.

The domain gateway architecture is designed for the ATMI application administrator, who makes services in other domains available to application programmers. The existence of applications within distinct domains is, however, totally transparent to the application programmers. They can use ATMI programming paradigms to request services offered in other domains as if they were services offered within the local application.

The ATMI application administrator enables remote domains to access a subset of *Local Services*. This subset is called a *Local Domain*. The local domain helps the administrator provide secure “views” of the application.

The eAM gateway communicates between independent Logical Units (LUs) using LU6.2 sessions and conversations. It adds multi-domain connectivity, bidirectional request/response, and conversations between ATMI platforms and SNA-based applications. It provides access to APPC applications based on SNA as well as inter-system communication with CICS/ESA. When accessing CICS/ESA systems, the eAM acts as a CICS/ESA region, capable of supporting the following sync-level 2 functions:

- Bidirectional request/response and conversational service requests
- Peer support for CICS/ESA and support for multiple remote MVS regions
- Event monitoring and reporting
- Both static and dynamic configuration support where allowed by SNA
- Automatic data conversion between UNIX and host formats
- Security through Access Control Lists on the local domain
- User and password security for CICS/ESA and IMS communications

The eAM Architecture

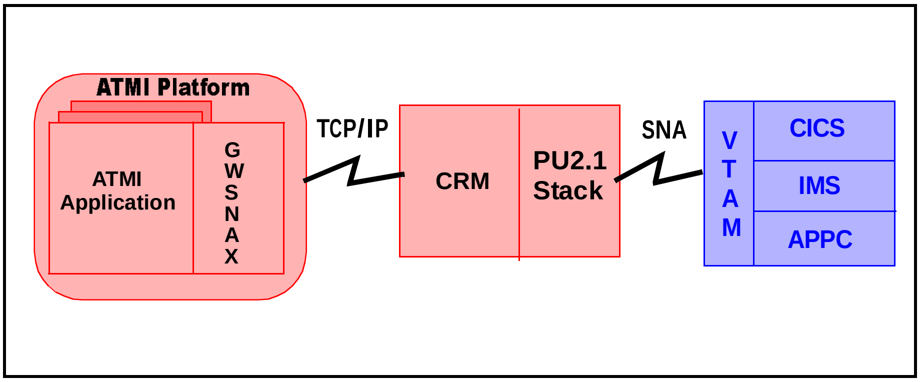
BEA eLink Adapter for Mainframe (eAM) is composed of two major components that can be configured to provide SNA solutions, the eAM gateway and the Communications Resource Manager. These components provide a bidirectional link enabling ATMI platforms such as BEA Tuxedo, eLink Platform, or WebLogic Enterprise to interact with IBM mainframe applications as either a client or server, using a CICS/ESA and IMS implicit LU6.2, or any IBM-supported Application Program-to-Program Communication (APPC) or CICS/ESA interface. Figure 1-1 illustrates the eAM architecture.

The eAM gateway (GWSNAX) runs in an ATMI platform environment and the Communications Resource Manager (CRM) can run in the native UNIX/NT environment or be distributed to the mainframe as a VTAM application. The CRM may also be distributed to another UNIX/NT system, separate from the ATMI platform.

Generally, the eAM domain is like other ATMI domain gateways. It uses the `DMADM` and `GWADM` servers for administration Within the eAM system, additional servers and processes support peer CICS/ESA and IMS connectivity and sync level 2.

Note: An ATMI platform is required, and a third-party SNA stack is required if the CRM is not installed on a mainframe. These products are sold separately.

Figure 1-1 eAM Architecture



The eAM Gateway

The eAM gateway (GWSNAX) is an ATMI domain gateway that communicates with the CRM. The eAM gateway processes ATMI-to-mainframe requests and responses in conjunction with the CRM. Requests coming from the mainframe are mapped to ATMI services while requests originating in ATMI are mapped to mainframe programs that can be executed using a CICS DPL or DPT application, or started from an IMS queue.

Communications Resource Manager

The CRM runs as a separate native process. It enables APPC conversations and DPL protocols to flow into and out of the ATMI environment. The CRM obtains its configuration from the gateway. If the gateway is running on a platform other than the one on which the CRM is running, the CRM should be started before the ATMI platform is started so it will be monitoring the address specified in the gateway configuration.

A properly configured SNA protocol stack is required for the CRM to communicate with a mainframe, unless the CRM is running independently of the ATMI environment (distributed mode).

2 Configuring the System

To enable your eLink Adapter for Mainframe (eAM), proper configuration based on your system's architecture is required. This section covers the following topics:

- Preparing for Configuration
- Configuring the eAM Gateway with the CRM

Note: All references to ATMI files, functions, and documentation apply to Tuxedo, eLink Platform, and WebLogic Enterprise files, functions, and documentation.

Preparing for Configuration

Before you can properly configure your eAM gateway with the CRM, you must complete the following prerequisites:

- Determine Your System Architecture
- Configure the Local Host
- Configure the Remote Host

Determine Your System Architecture

To determine your system's architecture, you must determine the location of the eAM components in that architecture.

eAM Components

The following basic components of the eAM system are factors in configuring your system:

- eAM gateway (GWSNAX)

  The eAM gateway is the transactional SNA gateway. It is implemented as an ATMI domain gateway and uses the ATMI environments. The gateway communicates over a Transmission Control Protocol/Internet Protocol (TCP/IP) connection.

- CRM

  The Communications Resource Manager (CRM) communicates with the SNA network, normally using an SNA stack. It communicates over the gateway with eAM clients over a Transmission Control Protocol/Internet Protocol (TCP/IP) connection.

- SNA stack

  The stack is vendor-supplied software that provides connectivity to an SNA network.

System Architecture

Your system architecture will reflect one of the following basic eAM configurations:

- Local Configuration
- Distributed Configuration—CRM on OS/390 Host
- Distributed Configuration—CRM on UNIX/NT Platform

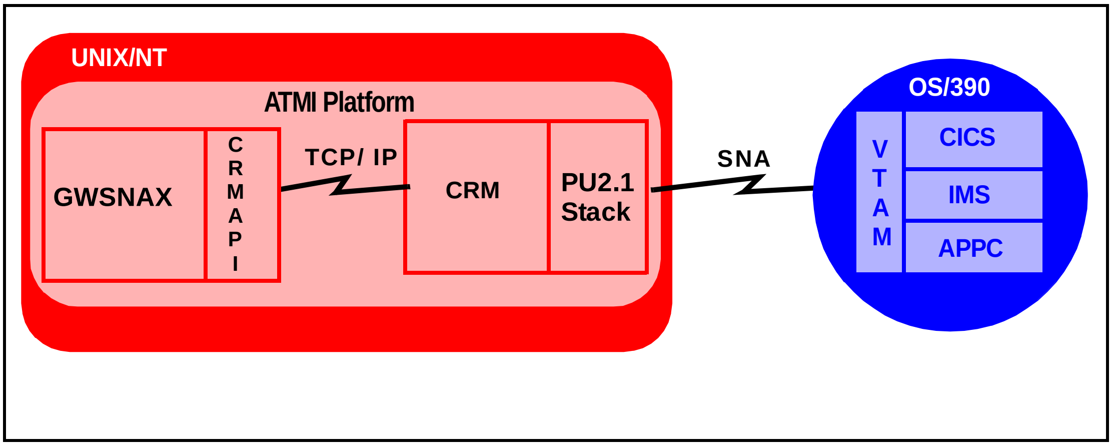
Local Configuration

Local configuration combines the ATMI platform, the eAM gateway, CRM, and SNA stack (PU2.1 server) on the same UNIX or Windows NT platform, as shown in Figure 2-1. It features the widely used TCP/IP connectivity between the eAM gateway and CRM, giving a high-performance communications interface. On the mainframe side, the CRM uses a stack to communicate over a System Network Architecture (SNA) interface with the host system. This configuration allows you to:

- Connect to an existing SNA network.
- Consolidate the ATMI platform and eAM systems on the same platform.
- Avoid having to deploy any subsystems to the host.

Note: A one-to-one relationship exists between the eAM gateway and CRM. The eAM gateway cannot be configured to handle multiple CRM processes.

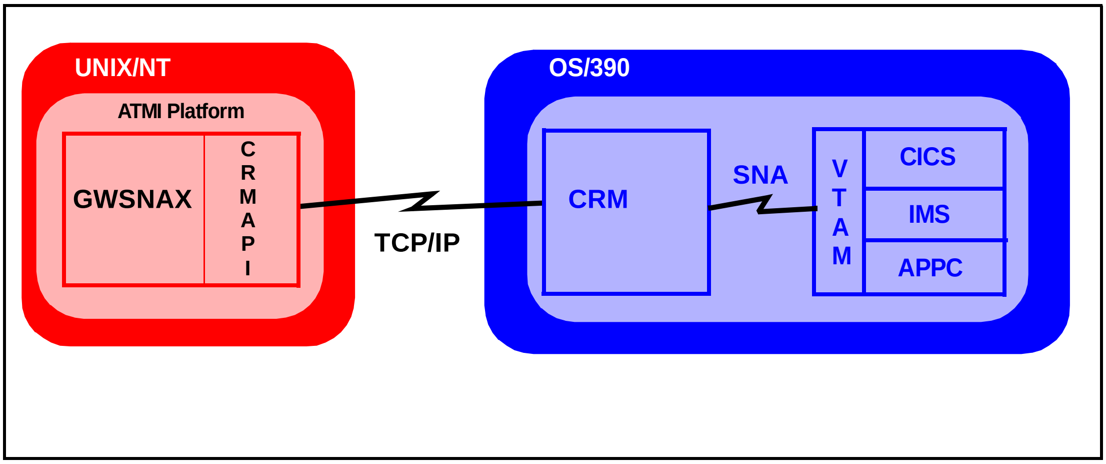
Figure 2-1 Local eAM Configuration on UNIX/NT Platform



Distributed Configuration—CRM on OS/390 Host

This distributed configuration deploys the CRM to the OS/390 host system as shown in Figure 2-2. It employs TCP/IP connectivity with the host, eliminating the need for a local SNA stack. This configuration provides a faster network interface and is less complex than the local configuration.

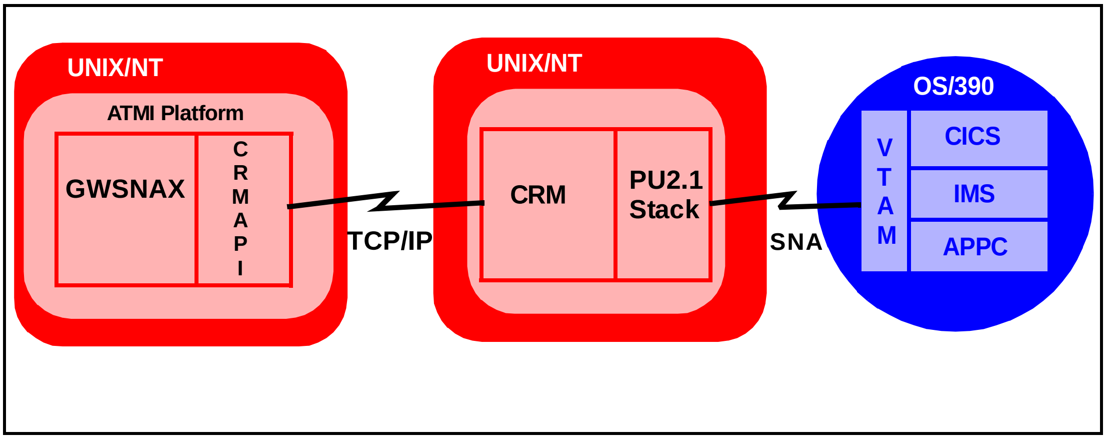
Figure 2-2 Distributed CRM on OS/390 Platform



Distributed Configuration—CRM on UNIX/NT Platform

This distributed configuration deploys the CRM and stack to a UNIX or Windows NT platform, as shown in Figure 2-3. It employs the TCP/IP connectivity between the eAM gateway and CRM, as well as the SNA connectivity to the host. This configuration allows you to use multiple stacks from different stack vendors. Also, on the ATMI platform side, you have a greater variety of UNIX/NT-based platform manufacturers to choose from.

Figure 2-3 Distributed CRM on UNIX/NT Platform



Configure the Local Host

Ensure that the local host is prepared to conduct operations with the remote host by completing the following task:

- Configure the Local LU for the appropriate stack.

Refer to the *BEA CRM Administration Guide* for more information about this task.

Configure the Remote Host

Ensure that the remote host is prepared to conduct operations with the ATMI local domain by completing the following tasks:

1. Configure the Remote LU.
2. Complete cross-platform definitions, if necessary.
3. Activate the connection between the remote host and the local host.

Refer to the *BEA CRM Administration Guide* for more information about these tasks.

Configuring the eAM Gateway with the CRM

The following list summarizes the tasks that must be completed to configure the eAM gateway (GWSNAX):

1. Edit the `DMTYPE` file.
2. Edit the `UBBCONFIG` file and load to create the binary.
3. Edit the `DMCONFIG` file and load to create the binary.
4. Start the CRM.
5. Start the ATMI servers.

Step 1: Edit the DMTYPE File

The `DMTYPE` file is an ASCII file. Use any text editor to edit this file.

1. Insert the following line in the `DMTYPE` file located in the `$TUXDIR/udataobj` directory:

- For UNIX:

```
SNAX::::
```

- For Windows NT:

```
SNAX;;;;
```

2. Ensure that the `$TUXDIR/udataobj/DMTYPE` file exists prior to editing the `DMCONFIG` file. See `dmloadcf` in Appendix A, "Reference Pages"for more information.

Step 2: Edit the UBBCONFIG File

The `UBBCONFIG` file is an ASCII file that can be edited with any text editor. To edit the `UBBCONFIG` file, complete the following tasks:

1. Create a `UBBCONFIG` file for each application. Refer to the Configuration section in the appropriate ATMI platform product documentation for more specific information about the `UBBCONFIG` file.
2. Establish a new gateway configuration or modify an existing one by defining the domain and gateway administrative servers to the ATMI system in the `UBBCONFIG` file.
3. If the CRM is to run as an ATMI server, add a CRM entry to the `*SERVERS` section of the `UBBCONFIG` file. For a description, refer to the *BEA CRM Administration Guide*.

Note: If the CRM is started as an ATMI process, it must precede the `GWSNAX` entry in the `UBBCONFIG` file.

4. Establish the eAM gateway by adding an entry to the `*SERVERS` section of the `UBBCONFIG` file. For a description, refer to the `GWSNAX` command in Appendix A, "Reference Pages." The following gateway features may be enabled in the `UBBCONFIG` file:
   - Data transformation
   - Bypassing user ID mapping
   - Encryption
   - Authentication
5. Refer to the appropriate ATMI platform documentation for instruction for using `tmloadcf` to load the `UBBCONFIG` file.

Figure 2-4 Sample `UBBCONFIG` File Entries Specifying CRM as an ATMI Server

```
*GROUPS

   SNAGRPLMID=mysys
        GRPNO=4


   LOCGRPLMID=mysys
        BRPNO=5
```

```
*SERVERS

   DEFAULT:CLOPT = “-A”


   DMADM SRVGRP=LOCGRP
        SRVID=14


   GWADM SRVGRP=SNAGRP
        SRVID=14
        RE PLYQ=Y
        RESTART=N
        GRACE=0


   SNACRM SRVGRP=SNAGRP
        SRVID=15
        CLOPT=“-A--//dalhps2:4452 SNAGRP”
        RESTART=Y
        RCMD=rstsnagrp
        GRACE=120
        MAXGEN=2


   GWSNAX SRVRGRP=SNAGRP
        SRVID=16
        RQADDR=”SNADOM”
        REPLYQ=N
        RESTART=Y
        RCMD=rstsnagrp
        GRACE=120
        MAXGEN=2
```

Step 3: Edit the DMCONFIG File

The configuration specified in the `DMCONFIG` file controls much of the operation of the eAM gateway (GWSNAX). A sample of this file is provided in the installation directory of your eAM product software.

Note: Because eAM may be installed on a variety of platforms, the procedures in this section make only general references to command entries. Many steps show UNIX command examples. Be sure to use the proper syntax for your platform when making command-line entries.

1. Verify that the eAM product software is installed and accessible to your text editor.
2. Verify that you have file permission to access the install directory and modify the sample `DMCONFIG` file.
3. Set each of the parameters of the `DMCONFIG` file as described in the following sections and load the `DMCONFIG` file. Refer to the appropriate ATMI documentation for instruction for using `dmloadcf` to load the `DMCONFIG` file.

   a. Update the `*DM_LOCAL_DOMAINS` Section.

      This section identifies local domains and their associated gateway groups. The section must have an entry for each gateway group (Local Domain). Entries have the form:

      `LDOM` required parameters {optional parameters}

      In this entry, `LDOM` is an identifier value used to name each local domain. For a full description of the optional and required parameters, see `DMCONFIG` in Appendix A, "Reference Pages."

      For each `LDOM` entry, the value of the `TYPE` parameter distinguishes this gateway from other gateway types. Currently, `SNAX` replaces the value `SNADOM` used in previous releases. The parameter entry takes the form:

      ```
      TYPE={SNAX | OSITP | TDOMAIN}
      ```

      Select the value `TYPE=SNAX` for the `LDOM` entry.

   b. Update the `*DM_REMOTE_DOMAINS` Section.

      This section identifies the known set of remote domains and their characteristics. Entries have the form:

      `LDOM` required parameters

      In this entry, `RDOM` is an identifier value used to identify each remote domain known to this configuration. For a full description of the required parameters, see `DMCONFIG` in Appendix A, "Reference Pages."

For each RDOM entry, the value of the TYPE parameter indicates that the remote domain communicates using the SNA protocol. The parameter entry takes the form:

```
TYPE={SNAX | OSITP | TDOMAIN}
```

Select the value TYPE=SNAX for the RDOM entry.

c. Add the *DM_SNACRM Section.

Note: *DM_SNACRM, *DM_SNASTACKS, and the *DM_SNALINKS sections have replaced the *DM_SNADOM section used in previous releases of eAM.

Note: Any changes to the *DM_SNACRM, *DM_SNASTACKS, or *DM_SNALINKS sections require a cold start for the eAM domain. If you do not cold start the eAM domain, an error will occur on domain start-up indicating cold start required for the configuration change.

The *DM_SNACRM section provides three keywords used to identify the CRM that provides ATMI transaction semantics in a given domain and its partners. Entries have the general form:

<CRMName> parameters

In this entry, <CRMName> is the locally known name of this SNACRM definition to be used when referencing this SNACRM in subsequent sections. This name is an ASCII string 1-30 characters in length. The parameters are the keyword/value pairs that make up the definition. All keywords are required for a valid SNACRM definition. Keywords can be in any order.

- LDOM=<LocalDomainName> (Required)

  LDOM associates this SNACRM with a defined local domain. <LocalDomainName> is the reference to an entry in the *DM_LOCAL_DOMAINS section. This name is an ASCII string 1 to 30 characters in length. This parameter is required. This parameter has no default.

- SNACRMADDR=<HexSocketAddress> (Required)

  SNACRMADDR provides the socket address the domain gateway uses to communicate with the SNACRM. This address represents the machine and port where the CRM runs. In a local configuration, this address is the local platform. In a distributed configuration, this address is a remote platform.

This address must be used on the `SNACRM` command line. This parameter is required and has no default.

`<HexSocketAddress>` is a TCP/IP address using `//hostname:port_addr` or the `sockaddr_in` format of family, port, address:

`<0xFFFFPPPPAAAAAAAA>`

In this entry, arguments and options are defined in the following way:

`FFFF` is the hex value of the protocol family, always `0x0002` for the INET family.

`PPPP` is the hex value of an unused TCP/IP port.

`AAAAAAAA` is the hex value of the IP address for the machine running the `SNACRM.`

Therefore, if the CRM was running on a machine named `myhost` with an IP address of `206.189.43.13`, and you wanted to use port 6000 for the CRM, then `SNACRMADDR` would be:

`//myhost:6000` or `0x00021770CEBD2B0D`

- `NWDEVICE=<Device Name>` (Required)

  `<Device Name>` is the logical name used to access the network. For example:

  `/dev/tcp`

d. Add the `*DM_SNASTACKS` Section.

The `DM_SNASTACKS` section provides five keywords that identify the third-party SNA stack that should be used for connections established between a given domain and its partners. Entries have the general form:

`<StackReference>` parameters

In this entry, `<StackReference>` is the locally known name of this stack definition and it is used when referencing this stack in subsequent sections. This name is an ASCII string 1-30 characters in length. The parameters are the keyword/value pairs that make up the definition. Keywords can be in any order. All keywords are required for a valid stack definition.

- `LOCALLU=<LocalLUAlias>` (Required)

  `LOCALLU` provides a reference to an LU alias defined in the third-party SNA stack. `<LocalLUAlias>` is the name used to identify the local LU definition as specified by the third-party SNA stack configuration. This name represents the end node for an LU6.2 connection. The value for this parameter is an ASCII string, 1-8 characters in length. This parameter is required. This parameter has no default. The third party SNA stack requires a corresponding definition for a local LU.

- `LTPNAME=<LocalTransactionProgramName>` (Required)

  `LTPNAME` identifies the inbound transaction programs that are serviced by any `SNACRM` using this stack definition. `<LocalTransactionProgramName>` is the name used to identify inbound transaction programs for which an attach will be accepted. The only useful value is an asterisk that indicates all inbound attaches will be accepted. This parameter is required. This parameter has no default. Partial TP names are not supported. The third-party SNA stack requires a corresponding definition for inbound TP names.

- `SNACRM=<CRMName>` (Required)

  `SNACRM` provides a name to which the associated `SNACRM` definition is referenced. `<CRMName>` is the name used to associate the `*DM_SNACRM` definition with this `*DM_SNASTACKS` entry. The value for this parameter is an ASCII string, 1-30 characters in length. This parameter is required. This parameter has no default.

- `STACKPARMS=<parameters required for third-party sna stack>` (Required)

  `STACKPARMS` provides a method for the domain gateway to pass any required parameters to the third party SNA stack. The `<parameters required for third-party sna stack>` is an ASCII string, 1-128 characters in length. Currently, the only value used is the TCP/IP hostname for the machine running the third-party SNA stack. This parameter is required. This parameter has no default.

- `STACKTYPE={hp60 | ibm60 | spx62 | sun91 | ms40 | vtm28}`

  This option is used to indicate which vender SNA stack is being used. It is also used to determine the names of specific eAM system libraries. Because of this, it is essential that the value of this option be coded correctly. These values are mapped to the equivalent eAM system library.

e. Add the `*DM_SNALINKS` Section.

   The `*DM_SNALINKS` section provides 11 key words that define the SNA Link information required by domains of type SNA. Entries have the general form:

`<Link Name>` parameters

In this entry, `<Link Name>` is the indentifier value used to identify the connection between a local domain (`LDOM`) and a remote domain (`RDOM`). This name is an ASCII string 1-30 characters in length. The parameters are the keyword/value pairs that make up the definition. Keywords can be in any order.

- `STACKREF=<Stack Reference>` (Required)

  This required parameter defines the stack that will be used for establishment of this link. The `STACKREF` string is the tag that was used in the corresponding definition established in the `*DM_SNASTACKS` section.

- `RDOM=<name>`

  Each link defines a connection between an ATMI system application domain and a remote system connected with an SNA network. The remote system is, in ATMI terms, a remote domain. The `RDOM` option associates the link with a remote domain. This remote domain must have been configured with the `TYPE=SNAX` option. The `RDOM` name should match an `RDOM` value previously identified in the `*DM_REMOTE_DOMAINS` section.

- `LSYSID=<name>`

  `LSYSID` is the four-character identifier for this link. This should match the connection ID in the CICS/ESA resource definition used by a partner CICS/ESA to communicate to the `SNACRM` across this link. If you are using the macro definition, it is a four-character name on the `SYSIDNT` option of the `DFHTCT` macro.

- `RSYSID=<name>`

  `RSYSID` is the four-character remote sysid of the partner. Typically it is the sysid of a CICS/ESA region, but could also be the subsystem ID of an IMS control region. This parameter must match the actual sysid of the remote partner. This name is the `SYSIDNT` of the `DFHSIT` or the value in the CICS/ESA start-up overrides

- `RLUNAME=<name>` (Required)

  The `RLUNAME` value represents an alias known to the third-party SNA stack that resolves to a VTAM `netname` for the remote application. This remote application is most likely the VTAM `applid` for a CICS/ESA region, however it could also be an APPC/MVS LU defined for use with IMS. The value must be unique within the SNA network. The value *name* should be

1-8 characters. This parameter is required. This parameter has no default. The third-party stack configuration requires a matching definition.

- `MODENAME=<name>` (Required)

  `MODENAME` is VTAM mode entry defined to the third-party SNA stack. For a CICS/ESA link, this entry must be compatible with the session definition or profile entry for the corresponding connection. For an IMS connection, this entry must be compatible with the `DLOGMOD` entry on the LU definition used to access the IMS scheduler. The value *name* should be 1-8 ASCII characters. This parameter is required. This parameter must match the third-party SNA stack configuration and must be compatible with the corresponding entries defined to VTAM and/or CICS/ESA.

- `SECURITY={LOCAL | IDENTIFY | VERIFY | PERSISTENT | MIXIDPE}`

  `SECURITY` specifies the security setting in CICS/ESA connection resource definition. It identifies the level of security enforced under CICS/ESA by the external security manager. Legal values are `LOCAL`, `IDENTIFY`, `VERIFY`, `PERSISTENT` or `MIXIDPE`. The default setting is `LOCAL`. `PERSISTENT` and `MIXIDPE` identify the setting in the remote connection definition, but are identical to the `VERIFY` option in this release of eAM.

- `MAXSESS=<number>`

  This number represents the maximum number of sessions that can be concurrently acquired on this link. It must be greater than or equal to four, and less than or equal to the maximum number of sessions that can be configured by the SNA stack. The actual number of concurrent sessions is determined by both system configurations to be the lowest maximum number of sessions allowed by either system.

- `MINWIN=<number>`

  This value is the minimum number of contention winners. Typically, this value is half the `MAXSESS` value. This number added to all CICS/ESA session definition winner numbers for the connection should be equal to the `MAXSESS` value.

- `STARTTYPE={AUTO|COLD}`

  This option sets the recovery mode for transactional links. When set to `AUTO`, the system restarts using configuration and link data recovered from the transaction log. When set to `COLD`, the system uses configuration data taken from the current `DMCONFIG` file and loses any in-flight link data. Changing

DMCONFIG file parameters and performing an AUTO start results in a message warning that changed parameters are ignored until the next cold start.

- MAXSYNCLVL={0 | 1 | 2}

  This value represents the maximum sync-level conversation that can be supported on this link. The default is sync-level 2. If the installation is not licensed for sync-level 2, this parameter must be set to 0 or 1 for the link to be established. Transaction support is only available at sync-level 2.

  Sync-level 0
  A value of zero (0) means this link is non-transactional. No synchronization is maintained. This value can be used for sending and receiving messages from IMS via the APPC/MVS transparency interface.

  Sync-level 1
  Allows sync-level 0 capabilities as well as support for SYNCONRETURN *Distributed Program Link (DPL)* with CICS/ESA systems (outbound ATMI tpcall() requests with TPNOTRAN).

  Sync-level 2
  Supports all sync-level 0 and sync-level 1 features for systems able to exchange logs and compare states. In addition, full syncpoint synchronization at sync-level 2 is supported.

Caution: If you set MAXSYNCLVL=2 or make no entry for this parameter (that is, accept the default) without having installed the eAM software licensed for that level, the system configuration automatically reverts to sync-level 1 and an error message is sent to the error log. To clear that error message, you must either reset the MAXSYNCLVL parameter to an appropriate value or purchase and install the correct software.

f. Update the *DM_LOCAL_SERVICES Section.

   The *DM_LOCAL_SERVICES section provides information on the services exported by each local domain. Entries have the general form:

   <Local Service Name> parameters

   In this entry, <Local Service Name> is the local name of the exported service. This name is an ASCII string 1-15 characters in length. The parameters are the keyword/value pairs that make up the definition. Keywords can be in any order. For a full description of parameters, see DMCONFIG in Appendix A, "Reference Pages."

- `RNAME=<name>` (Required)

  The `RNAME` option is the local-service name imported from a remote CICS/ESA region. This name is used by the CRM to select a local service.

  When the `RNAME` specifies an alternate mirror transaction identifier for explicit attachment for inbound DPL requests, it must be a combination of the alternate mirror `TRANSID` and a CICS/ESA program name in the following format:

  `RNAME=AAAA:BBBBBBBB`

  In this statement, the arguments and options are defined in the following way:

  `AAAA` is a 1-4 character alternate mirror TRANSID.

  `BBBBBBBB` is a 1-8 character CICS/ESA program name.

  The colon is required to indicate the `TRANSID`/program name combination. The `TRANSID` must be composed of acceptable CICS/ESA characters:

  `A-Za-z0-9$@#./-_%&Q¢?!|”=,;<>`

  Refer to “Special Treatment of TRANS ID for DPL” in “Application to Application Programming Considerations.”

g. Update `*DM_REMOTE_SERVICES` Section.

  The `*DM_REMOTE_SERVICES` section provides information on services “imported” and available on remote domains. Entries have the general form:

  `<Remote Service Name>` parameters

  In this entry, `<Remote Service Name>` is the name used by the local application for a particular remote service. This name is an ASCII string 1-15 characters in length. The parameters are the keyword/value pairs that make up the definition. Keywords can be in any order. For a full description of parameters, see `DMCONFIG` in Appendix A, “Reference Pages.”

- `FUNCTION={APPC | DPL}`

  The `FUNCTION` option has been added to allow outbound ATMI service requests to map to APPC transaction programs or CICS/ESA DPL programs. The default value is `APPC`.

- `RNAME=<name>`

The `RNAME` option is the name of the host TP_NAME. For non-CICS/ESA systems, this name can be up to 64 characters in length. For CICS/ESA systems, this name is the transaction ID for `FUNCTION=APPC` and the program name for `FUNCTION=DPL` requests. CICS/ESA trans-id names cannot exceed four characters and CICS/ESA program names cannot exceed eight characters. The `RNAME` option must observe these requirements.

When the `RNAME` specifies an alternate mirror transaction identifier for explicit attachment to outbound DPL requests, it must be a combination of the alternate mirror `TRANSID` and an advertised remote CICS/ESA program name in the following format:

```
RNAME=AAAA:BBBBBBBB
```

In this statement, the arguments and options are defined in the following way:

`AAAA` is a 1-4 character alternate mirror TRANSID.

`BBBBBBBB` is a 1-8 character CICS/ESA program name.

The colon is required to indicate the `TRANSID`/program name combination. The `TRANSID` must be composed of acceptable CICS/ESA characters:

```
A-Za-z0-9$@#./-_%&Q¢?!|”=,;<>
```

Refer to “Special Treatment of TRANS ID for DPL” in “Application to Application Programming Considerations.”

Step 4: Start the CRM

If the CRM is run in distributed mode or from the command line, it must be started independently of the ATMI processes. Start the CRM in one of the following ways:

- For MVS, use the JCL that was modified during the installation process.
- For all other distributed configurations, use the `SNACRM` command in the following format:

  ```
  SNACRM [parameters] <HexSocketAddress> <group name>
  ```

Refer to `SNACRM` in Appendix A, “Reference Pages” for more information about this command.

Step 5: Start the ATMI Servers

Perform a `tmboot` as described in the appropriate ATMI platform documentation to start the ATMI servers. If it is already running, perform a `tmshutdown` and `tmboot`.

3 Verifying the Software

After installing and configuring the eLink Adapter for Mainframe (eAM) software, verify the operational integrity of the environment by running a sample application on a simple server in client/server transaction scenarios. This process employs programs available in your product software libraries.

Note: All references to ATMI files, functions, and documentation apply to Tuxedo, eLink Platform, and WebLogic Enterprise files, functions, and documentation.

This section covers the following topics:

The sample applications are located in the simple server library (`eLink/sna/simpapp`). The simple server passes a string from the client to the server.

The CICS/ESA programs may run as either DTP or DPL processes, and as either servers or clients. The simple server may also run in either transactional or non-transactional mode. In the transactional mode, these scenarios verify that the sync-level 2 protocol is established between the two application environments.

When the client runs as an ATMI client, the server runs as a CICS/ESA host. Enter a text string in lower-case letters with command arguments. The CICS/ESA server converts the lower-case letters to upper-case letters and re-displays the text string.

When the client runs as a CICS/ESA client, the server runs as an ATMI server. Again, enter a text string in lower-case letters. The ATMI server converts the text string into a mirror image and displays the string as reversed letters.

Note: The verification process is intended for the CICS/ESA environment only—between ATMI applications and CICS/ESA applications. If your ATMI applications operate in other environments, you must create your own verification process. See the "APPC/IMS Programming Considerations" section.

Building Verification Tests

Build the verification test to run in two domains, the ATMI local domain and the CICS/ESA remote domain. The executables in each domain are different. The following sections discuss how to build these executables.

Building ATMI Platform Executables

To build the ATMI platform executables, perform the tasks that are described in detail in the following sections:

- Modify the UBBCONFIG File
- Execute the tmloadcf Command
- Modify the DMCONFIG File
- Execute the dmloadcf Command
- Modify the Environment Files
- Build the Server
- Build the Client

Modify the UBBCONFIG File

Modify the UBBCONFIG file to include the name of the ATMI simple server in the following way:

```
GROUPS
GROUP3          LMID=sna GRPNO=3

SERVERS
mirrorsrv       SRVGRP=GROUP3 SRVID=1 RQADDR=MIRR1 REPLYQ=Y

SERVICES
MIRROR
```

Enable Transactional Services

If you plan to run the transactional version of the verification process, enable the TLOGDEVICE comment in the Machine section to point to a valid DTP transaction log. To run transaction examples, create the DTP transaction log named on the UBBCONFIG TLOGDEVICE with the ATMI platform bulletin modification interpreter tmadmin. Refer to the appropriate ATMI platform documentation for option descriptions.

The application server GROUP3 in the Groups section must point to a valid transaction manager server. For example:

```
GROUPS
GROUP3          LMID=sna GRPNO=3 TMSNAME=tsttms TMSCOUNT=2

SERVERS
mirrorsrv       SRVGRP=GROUP3 SRVID=1 RQADDR=MIRR1 REPLYQ=Y

SERVICES
MIRROR
```

Execute the tmloadcf Command

Execute the ATMI tmloadcf command to parse the UBBCONFIG file and create a binary version of the file. Refer to the appropriate ATMI platform documentation for option descriptions.

For example:

```
tmloadcf UBBCONFIG
```

Respond to the prompts as the command executes.

Modify the DMCONFIG File

The DMCONFIG file must contain both local and remote definitions for the simple server.

Note: A sample DMCONFIG file is included with the simple server.

Listing 3-1 Sample DMCONFIG File

```
*DM_LOCAL_SERVICES
#The Atmi reverse string server
MIRROR       LDOM="simpsnad"
       CONV=N
       RNAME="MIRRORSERV"
       INBUFTYPE="STRING"
       OUTBUFTYPE="STRING"
DOUBLEMIRROR
       CONV=N
       RNAME="MIRRDPLS"
       INBUFTYPE="STRING"
       OUTBUFTYPE="STRING"
*DM_REMOTE_SERVICES
#The CICS upper-case DTP and DPL servers
SIMPDPL      AUTOTRAN=N
       LDOM="simpsnad"
       RDOM=SNAG1
       CONV=N
       RNAME="TOUPDPLS"
       INBUFTYPE="STRING"
       OUTPBUFTYPE="STRING"
       FUNCTION="DPL"
SIMPDTP      AUTOTRAN=N
       LDOM="simpsnad"
       RDOM=SNAG1
       CONV=N
       RNAME="DTPS"
       INBUFTYPE="STRING"
       OUTPBUFTYPE="STRING"
       FUNCTION="APPC"
```

In the preceding DMCONFIG file example, both instances of the LDOM name correspond to the SNA domain name in the DM_LOCAL_DOMAINS section. The server is a request/response server.

In the DM_LOCAL_SERVICES section, the RNAME=”MIRRORSERV” and RNAME="MIRRDPLS" values are the names passed from the CICS/ESA environment. MIRROR and DOUBLEMIRROR refer to the advertised services provided by the mirrorsrv server named in the UBBCONFIG file. The CONV=N definition indicates the protocol that is observed by the SNA domain, although the CICS/ESA client does not perform an ATMI tpcall.

In the DM_REMOTE_SERVICES section, the RNAME value identifies what is invoked in the CICS/ESA domain. For the Distributed Program Link (DPL) request, the RNAME equals the name of the program called (or for alternate mirror transaction identifiers, a TRANSID/program name combination). For the Distributed Transaction Processing (DTP) request, the RNAME equals the name of the transaction ID.

If you want to run transactional verification tests, you must enter a link definition MAXSYNCLVL=2 in the DM_SNALINKS section. If you want to run non-transactional DPL tests only, you must enter MAXSYNCLVL=1.

Execute the dmloadcf Command

Execute the ATMI dmloadcf command to parse the DMCONFIG file and create a binary version of the file. Refer to the appropriate ATMI platform documentation for option descriptions.

For example:

```
dmloadcf DMCONFIG
```

Respond to the prompts as the command executes.

Modify the Environment Files

Two types of files are provided with your eAM product software that can be used to define the application and/or machine environments for verification testing. If their equivalents do not already exist, modify the files provided and make them available to your system. The files are apps.env and <machine>.env.

The apps.env File

Modify the apps.env file and include it with the ENVFILE parameter in the MACHINES section of the UBBCONFIG file. The apps.env file looks like this:

Listing 3-2 The apps.env File

```
#=================================================================
# apps.env
#       Environment macros for atmi application testing.
#
# See also
#       See $(TOP)/Makefile for more information.
#
# @(#)SNA Devel apps/simpsna app.env 1.4 98/03/03 15:42:30
# Copyright 1997, BEA Systems, Inc., all rights reserved.
#-----------------------------------------------------------------
APPDIR=<Your application directory here>
TUXCONFIG=<Your Tux configuration here>
BDMCONFIG=<Your Domain configuration here>
TUXDIR=<Your Atmi directory here>
FLDTBLDIR32=<Your Atmi directory here>/lib
FIELDTBLS32=fmb.def
```

The <Machine>.env Files

Modify the `<machine>.env` file that is appropriate for your system:

- `aix.env`
- `hpux.env`
- `solaris.env`
- `ntenv.bat`
- `uss.env`
- `ENV (MVS)`

Each of these files is executable. Once you have modified the appropriate file for your system, execute it to export the machine environment variables. The files look like this:

Listing 3-3 The aix.env File

```
#=================================================================
# aix.env
#       Environment macros for AIX testing.
#
# See also
#      See $(TOP)/Makefile for more information.
#
# @(#)SNA Devel apps/simpsna aix.env 1.2 98/02/23 12:39:36
# Copyright 1997, BEA Systems, Inc., all rights reserved.
#-----------------------------------------------------------------
export APPDIR=<Your application directory bin here>
export TUXCONFIG=<Your atmi configuration qualified name here>
export BDMCONFIG=<Your atmi domain configuration qualified name
here>
export TUXDIR=<Your atmi product directory here>
export STACK=<Your stack product library here>
#example STACK=/opt/SUNWappc or /opt/sna/lib
export FLDTBLDIR32=$TUXDIR/lib
export FIELDTBLS32=fmb.def
export PATH=$APPDIR:$TUXDIR/bin:$PATH
export LIBPATH=$TUXDIR/lib:$LIBPATH:$STACK
```

Listing 3-4 The hpux.env File

```
#=================================================================
# hpux.env
#      Environment macros for HP-UX testing.
#
# See also
#      See $(TOP)/Makefile for more information.
#
# @(#)SNA Devel apps/simpsna hpux.env 1.3 98/02/23 12:38:34
# Copyright 1997, BEA Systems, Inc., all rights reserved.
#-----------------------------------------------------------------
export APPDIR=<Your application directory bin here>
export TUXCONFIG=<Your atmi configuration qualified name here>
export BDMCONFIG=<Your atmi domain configuration qualified name
here>
export TUXDIR=<Your atmi product directory here>
export STACK=<Your stack product library here>
#example STACK=/opt/sna/lib
export PATH=$APPDIR:$TUXDIR/bin:$PATH
export SHLIB_PATH=$APPDIR:$TUXDIR/lib:$SHLIB_PATH:$STACK
```

Listing 3-5 The solaris.env File

```
#=================================================================
# solaris.env
#      Environment macros for SOLARIS testing.
#
# See also
#      See $(TOP)/Makefile for more information.
#
# @(#)SNA Devel apps/simpsna solaris.env 1.3 98/02/23 12:39:05
# Copyright 1997, BEA Systems, Inc., all rights reserved.
#-----------------------------------------------------------------

export APPDIR=<Your application directory bin here>
export TUXCONFIG=<Your atmi configuration qualified name here>
export BDMCONFIG=<Your atmi domain configuration qualified name
here>
export TUXDIR=<Your atmi product directory here>
export STACK=<Your stack product library here>
#example STACK=/opt/SUNWappc
export FLDTBLDIR32=$TUXDIR/lib
export FIELDTBLS32=fmb.def
export PATH=$APPDIR:$TUXDIR/bin:$PATH
export LD_LIBRARY_PATH=$TUXDIR/lib:$LD_LIBRARY_PATH:$STACK
```

Listing 3-6 The ntenv.bat File

```
#================================================================
# ntenv.bat
#      Environment macros for WINDOWS NT testing.
#
# @(#)$Header
# Copyright 1998, BEA Systems, Inc., all rights reserved.
#-----------------------------------------------------------------
set APPDIR=<Your application directory bin here>
set TUXCONFIG=<Your tuxedo configuration qualified name here>
set BDMCONFIG=<Your tuxedo domain configuration qualified name
here>
set TUXDIR=<Your tuxedo product directory here>
set STACK=<Your stack product library here>
#example STACK=c:\ibmcs
set FLDTBLDIR32=%TUXDIR%\lib
set FIELDTBLS32=fmb.def
set PATH=%APPDIR%;%TUXDIR%\bin;%STACK%;%PATH%
```

Listing 3-7 The uss.env File

```
==============================================================
# uss.env
#      Environment macros for OS/390 USS testing.
#
# @(#)SNA $Header: /repos/sna/apps/simpsna/uss.env,v 1.2 2000/04/26
19:32:58 crount Exp $
# Copyright 2000, BEA Systems, Inc., all rights reserved.
#---------------------------------------------------------------
export APPDIR=<Your application directory bin here>
export EAMDIR=<Your eAM installation directory here>
export FLDTBLDIR32=$EAMDIR/lib
export FIELDTBLS32=fmb.def
export PATH=$APPDIR:$EAMDIR/bin:$PATH
export LIBPATH=$EAMDIR/lib:$LIBPATH
```

Listing 3-8 The ENV (MVS)File

```
*===============================================================
* mvs.env
*      Environment macros for running the SNACRM on MVS.
*
* @(#)SNA $Source: /repos/sna/apps/simpsna/mvs.env,v $
*         $Revision: 1.4 $   $Author: crount $
*         $Date: 2000/08/22 15:30:46 $   $State: Exp $ $Locker:  $
* Copyright 2000, BEA Systems, Inc., all rights reserved.

*--------------------------------------------------------------

APPDIR=<Your site high-level qualifier here>
*example APPDIR=BEA
FLDTBLDIR32=(DD:FLDTBL)
FIELDTBLS32=FMB
```

Build the Server

Use the ATMI platform `buildserver` utility to build the `mirrorsrv` server load module from the provided source file `mirrorsrv.c`. The source file contains two service entries, `MIRROR` and `DOUBLEMIRROR`, which will be advertised by the `mirrorsrv server.`

When executed, the `MIRROR` service receives a text string from the client, reverses the letters, and displays a mirror image of the input text string.

When executed, the `DOUBLEMIRROR` service receives a text string from the client, reverses the letters, and concatenates the reversed string to the forward image of the string.

The following command is an example of a command entry to invoke the `buildserver` utility:

```
buildserver -o mirrorsrv -f mirrorsrv.c -s MIRROR,DOUBLEMIRROR
```

Refer to the appropriate ATMI platform documentation for option descriptions.

Build the Client

Use the ATMI platform `buildclient` utility to build a client load module from the provided source file `toupclt`. When executed, the load module sends a lower-case text string to the server, which converts it to uppercase in several modes, causing different server scenarios to execute.

For example:

```
buildclient -o toupclt -f toupclt.c
```

Refer to the appropriate ATMI platform documentation for option descriptions.

Building CICS/ESA Executables

To build the CICS/ESA executables, perform the tasks that are described in detail in the following sections:

- Choose the Source Code Language
- Transfer the Source Code to the Host

- Translate CICS/ESA Verbs
- Compile the Translated Source File
- Create the Executable Object
- Configure the CICS/ESA Application
- View Connection and Session Status

Choose the Source Code Language

The CICS/ESA sample programs used for verification are unloaded during the installation of your eAM product software. These programs are available in two languages, COBOL and C. You must choose which language to use to build the CICS/ESA executable object code. (Your choice might be affected by the type of compiler available on your MVS host.)

You can identify the sample program names by their suffixes:

- `TOUPDTPS.c` is the C language program name
- `TOUPDTPS.cbl` is the COBOL language program name

Although the structures of the sample programs are different, they both perform the same function. The ATMI platform executable program that you build communicates with either.

Transfer the Source Code to the Host

Transfer the source code to the host by the method you prefer, for example FTP (File Transfer Program). The destination could be a sequential dataset or a PDS file. Table 3-1 lists the source code files provided for UNIX and MVS platforms. The UNIX filename extensions suggest the type of destination libraries into which the source code may be transferred.

Table 3-1 Source Code Filenames

UNIX Filename	MVS Member Name
BEACONN.RDO	BEACONN
BEASNA.RDO	BEASNA
MIRRDPLC.c or MIRRDPLC.cbl	MIRRDPLC
MIRRDTPC.c or MIRRDTPC.cbl	MIRRDTPC
TOUPDPLS.c or TOUPDPLS.cbl	TOUPDPLS
TOUPDTPS.c or TOUPDTPS.cbl	TOUPDTPS

Translate CICS/ESA Verbs

This step translates the `EXEC CICS` verbs into program `CALL` statements of the form required by the selected source language. The source is read from the SYSIN dataset. The translated source program is written to the `SYSPUNCH` dataset. The translator listing is written to the `SYSPRINT` dataset.

Different translator modules are provided for different source languages. There are also language-specific parameters for the translation step. Refer to the *IBM CICS/ESA Application Programming Guide* for additional translation options that might apply to your environment.

The translator modules are installed in the CICS/ESA load datasets. This is indicated in the following examples by the `CICSxxx.SDFHLOAD` entry, where `xxx` is the CICS/ESA release number.

COBOL Language Translator Example

The translator module name in the following example is `DFHECP1$`. The parameter `COBOL2` indicates that a source module containing COBOL II verbs is to be translated.

Listing 3-9 COBOL Language Translator Example

```
//TRN  EXEC PGM=DFHECP1$,
//     PARM=‘COBOL2,NOS,CICS’,REGION=256K
//STEPLIB     DD DSN=CICSXXX.SDFHLOAD,DISP=SHR
//SYSIN       DD DSN=YOUR.PDS(pgmname),DISP=SHR
//SYSPRINT    DD SYSOUT=*
//SYSPUNCH    DD DSN=&&SYSCIN,
//            DISP=(,PASS),UNIT=SYSDA,
//            DCP=BLKSIZE=400
//            SPACE=(400,(400,100))
```

C Language Translator example

The translator module name in Listing 3-10 is `DFHEDP1$`. The parameter `C` indicates that a source module containing C verbs is to be translated.

Listing 3-10 C Language Translator Example

```
//TRN  EXEC PGM=DFHEDP1$,
//     PARM=‘C,NOS,CICS’,REGION=256K
//STEPLIB     DD DSN=CICSXXX.SDFHLOAD,DISP=SHR
//SYSIN       DD DSN=YOUR.PDS(pgmname),DISP=SHR
//SYSPRINT    DD SYSOUT=*
//SYSPUNCH    DD DSN=&&SYSCIN,
//            DISP=(,PASS),UNIT=SYSDA,
//            DCB=BLKSIZE=400
//            SPACE=(400,(400,100))
```

Compile the Translated Source File

The next step is to compile the translated source file `&&SYSCIN`. This step generates the following program modules in preparation for the link-edit step:

- `TOUPDPLS` (CICS server DPL module)
- `TOUPDTPS` (CICS server DTP module using CICS APIs)
- `MIRRDPLC` (CICS client DPL module)
- `MIRRDTPC` (CICS client DTP module using CPI-C verbs)

COBOL Compiler Example

Listing 3-11 shows a COBOL compiler example.

Listing 3-11 COBOL Compiler Example

```
//COB  EXEC PGM=IGYCRCTL,REGION=GM,
//     PARM=‘NODYNAM,RENT,RES,APOST,MAP,XREF’
//STEPLIB     DD DSN=SYS2.COB2.COB2COMP,DISP=SHR
//SYSLIB      DD DSN=CICSXXX.SDFCOB,DISP=SHR
//SYSPRINT    DD SYSOUT=*
//SYSIN       DD DSN=&&SYSCIN,DISP=(OLD,DELETE)
//SYSLIN      DD DSN=&&LOADSET,DISP=(MOD,PASS),
//            UNIT=&WORK,SPACE=(400,(20,20))
//SYSUT1      DD UNIT-WORK,SPACE=(460,(350,100))
//SYSUT2      DD UNIT=WORK,SPACE=(460,350,100))
//SYSUT3      DD UNIT=WORK,SPACE=(460,(350,100))
//SYSUT4      DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT5      DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT6      DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT7      DD UNIT=&WORK,SPACE=(460,(350,100))
```

C Compiler Example

Compile and pre-link your C language source. The BEA eAM installation library contains C language compiler examples. Listing 3-12 shows a C/MVS compiler example.

In this example, the external storage variables must be re-entrant in the load module for the CICS/ESA environment. Additionally, the C language source contains long variable and function names. The pre-link step must be run to resolve the long names and create re-entrant variables from the compiled object. Perform the pre-link step, use the RENT option in the compile program, and use the compile output in the link-edit step.

Listing 3-12   C/MVS Compiler Example

```
//*---------------------------------------
//* COMPILE STEP:
//*---------------------------------------
//COMPILE EXEC PGM=CBC310,REGION=2M,COND=(7,LT,TRN),
//     PARM=('OPT(1),LONGNAME,RENT,SOURCE')
//STEPLIB     DD DSNAME=SYS1.SCEERUN,DISP=SHR
//            DD DSNAME=SYS1.SCBC3CMP,DISP=SHR
//SYSMSGS     DD DSNAME=SYS1.SCBC3MSG(EDCMSGE),DISP=SHR
//SYSIN       DD DSNAME=&&SYSCIN,DISP=SHR **FROM TRN STEP
//SYSLIB      DD DSNAME=SYS1.SCEEH.H,DISP=SHR
//            DD DSNAME=SYS1.SCEEH.SYS.H,DISP=SHR
//            DD DSNAME=CICSxxx.SDFHC370,DISP=SHR**CPIC
//            REQUIRED**
//SYSLIN      DD DSNAME=&&PLNKSET,UNIT=VIO,
//            DISP=(MOD,PASS),SPACE=(TRK,(3,3)),
//            DCP=(RECFM=FG,LRECL=80,BLKSIZE=&SYSLBLK)
//SYSPRINT    DD SYSOUT=*
//SYSOUT      DD SYSOUT=*
//SYSCPRT     DD SYSOUT=*
//SYSUT1      DD UNIT=VIO,SPACE=(32000,(30,30,)),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200
//SYSUT4      DD UNIT=VIO,SPACE=(32000,(30,30,)),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200
//SYSUT5      DD UNIT=VIO,SPACE=(32000,(30,30,)),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=12800
//SYSUT6      DD UNIT=VIO,SPACE=(32000,(30,30,)),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=12800
//SYSUT7      DD UNIT=VIO,SPACE=(32000,(30,30,)),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=12800
//SYSUT8      DD UNIT=VIO,SPACE=(32000,(30,30,)),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=12800
//SYSUT9      DD UNIT=VIO,SPACE=(32000,(30,30,)),
//            DCB=(RECFM=FB,LRECL=137,BLKSIZE=882
//SYSUT10     DD SYSOUT=*
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200
//SYSUT14     DD UNIT=VIO,SPACE=(32000,(30,30,)),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=128007
//PLKED EXEC PGM=EDCPRLK,COND=((7.LT,C),(7,LT,TRN)),
//            PARM='MAP'
//STEPLIB     DD DSN=SYS1.SCEERUN,DISP-SHR
//SYSMSGS     DD DSN=SYS1.SCEEMSGP(EDCPMSGE),DISP+SHR
//SYSLIB      DD DSN=&LE370HLQ..SCEECPP,DISP=SHR
//            DD DUMMY
//SYSIN       DD DSN=&&PLNKSET,DISP=(MOD,PASS)
//SYSMOD      DD DSN=&&LOADSET,DISP=(,PASS),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
```

```
//SYSOUT      DD SYSOUT=*
//SYSPRINT    DD SYSOUT+*
```

Create the Executable Object

The next step is to take the compiled source and create the executable object. You should load the resulting object module into the application library that is concatenated with the CICS/ESA region datasets.

In the following COBOL and C program examples, `SYSLIN` is the name of the file containing the compiled source concatenated with other necessary executables, including interfaces for the CICS API verbs, interfaces for CPI-C verbs, and interfaces for SAA resource recovery verbs (sync-level 2).

In Listing 3-13, the module is linked as re-entrant and marked with 31-bit mode address-ability. This action is required for the module `MIRRDTPC` which performs CPI-C and SAA Resource/Recovery requests.

You must change the `NAME` to that of the executable being generated.

Listing 3-13 COBOL Link-Edit Sample for TOUPDPLS, TOUPDTPS, and MIRRDPLC

```
//LKED    EXEC PGM=IEWL,   * ** LINKAGE EDITOR **
//             PARM=AMODE=31,RENT,
//             REGION=512K
//SYSPRINT DD SYSOUT=*
//SYSLIN       DD DSN=&&LOADSET,DISP=(OLD,DELETE)
//             DD *
INCLUDE SYSLIB(DFHECI)
ORDER DFHECI
NAME xxxxxxxx(R)
/*
//SYSLIB    DD DSN=CICSxxx.SDFHLOAD,DISP=SHR
//          DD DSN=CICSxxx.SDFHCOB,DISP=SHR
//          DD DSN=SYS1.SCEELKED,DISP=SHR
//          DD DSN=SYS1.SIGYCOMP,DISP=SHR
//SYSLMOD   DD DSN=application.loadlib,DISP=(SHR,PASS)
//SYSUT1    DD UNIT=VIOD,SPACE=(1024,(50,20))
//
```

Listing 3-14 COBOL Link-Edit Sample for MIRRDTPC

```
//LKED    EXEC PGM=IEWL,   * ** LINKAGE EDITOR **
//           PARM=AMODE=31,RENT,
//           REGION=512K
//SYSPRINT DD SYSOUT=*
//SYSLIN       DD DSN=&&LOADSET,DISP=(OLD,DELETE)
//           DD *
INCLUDE SYSLIB(DFHECI)
INCLUDE SYSLIB(DFHCPLC)
INCLUDE SYSLIB(DFHCPLRR)
ORDER DFHECI
NAME xxxxxxxx(R)
/*
//SYSLIB    DD DSN=CICSxxx.SDFHLOAD,DISP=SHR
//          DD DSN=CICSxxx.SDFHCOB,DISP=SHR
//          DD DSN=SYS1.SCEELKED,DISP=SHR
//          DD DSN=SYS1.SIGYCOMP,DISP=SHR
//SYSLMOD   DD DSN=application.loadlib(xxxxxxxx),DISP=(SHR,PASS)
//SYSUT1    DD UNIT=VIOD,SPACE=(1024,(50,20))
//
```

Listing 3-15 C Link-Edit Sample

```
//LKED     EXEC PGM=HEWL,REGION=4M,
//         PARM=’AMODE=31,RENT,
//         COND=((7,LT,C),(7,LT,PLKED),(7,LT,TRN))
//SYSLIB   DD DSN=CICSxxx.SDFHLOAD,DISP=SHR
//         DD DSN=SYS1.SCEELKED,DISP=SHR
//SYSLIN   DD DSN=&&LOADSET,DISP=(OLD,DELETE)
//         DD *INCLUDE SYSLIB(DFHELII)
               INCLUDE SYSLIB(DFHCPLC)
               INCLUDE SYSLIB(DFHCPLRR)NAME xxxxxxxx(R)
/*
//SYSLMOD  DD DSN=application.loadlib(xxxxxxxx),DISP=SHR
//SYSUT1   DD UNIT=VIOD,SPACE=(1024,(50,20))
//SYSPRINT DD SYSOUT=*
```

Configure the CICS/ESA Application

Your installed eAM software contains two sample files that can be used to configure the CICS application:

- The BEACONN file contains the CICS/ESA configuration parameters to the host system. These include connection definitions and session definitions.
- The BEASNA file contains the application definitions that enable you to perform the installation verification in the CICS/ESA environment. These definitions are required to run the installation verification. They include the program definitions, transaction definitions, and for the CPI-C example, the partner definition.

Caution: The BEACONN file should only be added to the CICS/ESA System Definition (CSD) file if no definitions currently exist. Check with your system administrator.

One method of adding the files is to use the batch utility program, DFHCSDUP. Listing 3-16 is an example of the Job Control Language (JCL) statements you can use to invoke DFHCSDUP as a batch program to add the BEASNA file:

Listing 3-16 JCL Example for Invoking DFHCSDUP

```
//YOURJOB JOB accounting info,name,MSGLEVEL=1
//STEP1 EXEC PGM=DFHCSDUP,REGION=512K,
//      PARM=’CSD(READWRITE),PAGESIZE(60),NOCOMPAT’
//STEPLIB DD DSN=CICSxxx.SDFHLOAD,DISP=SHR
//DFHCSD   DD UNIT=SYSDA,DISP=SHR,DSN=CICSxxx.DFHCSD
//SYSPRINT DD SYSOUT=A
//SYSIN    DD   DSN=YOUR.PDS(BEASNA),DISP=SHR
```

The definitions in the sample member use an eAM Resource Definition Online (RDO) Group name. You may want to add these definitions to an existing RDO group, or you might consider adding them to your CICS/ESA start-up list if you plan to use them often. (This automatically installs the group on start-up of the CICS/ESA region.) To add the groups to the start-up list, un-comment the following statements in the sample RDO.

```
ADD GROUP(BEACONN) LIST(**YOURLIST**)
ADD GROUP(BEASNA) LIST(**YOURLIST**)
```

In these arguments, `**YOURLIST**` is the name of your startup list.

To manually install the groups after start-up of the CICS/ESA region, issue the following commands from a CICS/ESA terminal session.

```
CEDA I GROUP(BEACONN)
```

```
CEDA I GROUP(BEASNA)
```

BEACONN File: Connection definition

The BEACONN file includes a sample connection definition.

Note: The sample connection definition achieves the minimum requirements for a connection over which the installation verification can be executed. Do not rely on it to provide optimal performance. Consult the *CICS/ESA Resource Definition Guide* for information about adding options not included in the sample.

The name of the sample connection definition is BEA, which is located under the installation group name BEACONN. It looks like this:

Listing 3-17 Sample Connection Definition in BEACONN File

```
DEFINE CONNECTION(BEA)      GROUP(BEACONN)
       DE(EAM EXAMPLE RDO CONNECTION)
       ACCESSMETHOD(VTAM)   PROTOCOL(APPC)
       NETNAME(**LOCALLU**)
       ATTACHSEC(LOCAL)     AUTOCONNECT(NO)
```

`**LOCALLU**` must be changed to the LU name of the SNA stack as known by VTAM. The `ATTACHSEC` option indicates the level of attach-time user security required for the connection. `LOCAL` is the simplest security. The authorization of the user is taken to be that of the link itself, relying on the authorization validation provided by the remote security utility. `AUTOCONNECT` indicates when the connection is acquired. `NO` is required. This entry means that CICS does not attempt to bind sessions when the connection is established by the stack.

Note: The eAM software must acquire the connection and negotiate the bind when the eAM software starts up.

Install the sample connection definition by putting it on the host in a separate group that does not contain existing connection. For example, use the `CEDA INSTALL` command:

```
CEDA INSTALL GROUP(BEACONN)
```

BEACONN File: Session Definition

The `BEACONN` file also includes a sample session definition. When placed on the remote host, it defines the logical links by which the ATMI platform local domain communicates with the remote host.

The name of the sample session definition is `BEATEST`, which is located under the installation group name `BEACONN`. It looks like this:

Listing 3-18 Sample Session Definition in BEACONN File

```
DEFINE SESSION(BEATEST)     GROUP(BEACONN)
       CONNECTION(BEA)
       DE(EAM EXAMPLE RDO SESSION)
       PROTOCOL(APPC)       AUTOCONNECT(YES)
       MODENAME(**MODE**) MAXIMUM(**MAX**,**MIN**)
```

`AUTOCONNECT` indicates how the activation of the session is to be negotiated. `YES` enables the CICS/ESA host to negotiate its own winner sessions when the connection is bound. Remember that the eAM software must acquire the connection instead of the CICS/ESA host. However, when the stack acquires the connection, it can only bind the number of sessions identified as its winners. Setting the `AUTOCONNECT` parameter to `YES` causes the host to bind winner sessions immediately when the connection is acquired. Otherwise, the host's outbound clients must wait for winner sessions to bind.

Replace `**MODE**` with either a CICS/ESA-supplied mode name, such as `SMSNA100`, or with your own defined mode name. If another set of session definitions exist for the BEA connection, this mode name must be unique among all sets defined to the connection. The mode name corresponds to A VTAM `LOGMODE` name.

The MAXIMUM option defines the total number of sessions in the set (**MAX**) and the total number of contention winner sessions (**MIN**). To verify the installation, the total number of winner sessions must include those for the host and the remote stack. The installation verification process allows both sides to execute as the client. The total number of local contention winner sessions plus remote contention winner sessions must equal the number of sessions. The local number of sessions must equal the remote number of sessions.

BEASNA File: Program Definition

The BEASNA file includes a sample program definition, shown in Listing 3-19. Replace the LANGUAGE variable **LANG** in the sample with either C or COBOL to identify the source type you have selected for the sample application.

Listing 3-19 Sample Program Definitions in BEASNA File

```
DEFINE  PROGRAM(MIRRDPLC)    GROUP(BEASNA)
        DE(EAM EXAMPLE CICS DPL CLIENT (MIRROR STRING))
        LANGUAGE(**LANG**)   DATALOCATION(ANY)

DEFINE PROGRAM(MIRRDTPC)     GROUP(BEASNA)
       DE(EAM EXAMPLE CICS DTP CLIENT (MIRROR STRING))
       LANGUAGE(**LANG**)    DATALOCATION(ANY)

DEFINE PROGRAM(TOUPDTPS)     GROUP(BEASNA)
       DE(EAM EXAMPLE CICS DTP SERVER (TOUPPER STRING))
       LANGUAGE(**LANG**)    DATALOCATION(ANY)

DEFINE PROGRAM(TOUPDPLS)     GROUP(BEASNA)
       DE(EAM EXAMPLE CICS DPL SERVER (TOUPPER STRING))
       LANGUAGE(**LANG**)    DATALOCATION(ANY)
```

BEASNA File: Remote Program Definition

The BEASNA file also contains a sample remote program definition, shown in Listing 3-20. The program definition is used by the CICS DPL client to identify the remote system and service for the DPL request. Replace the REMOTESYSTEM variable **CONNECTION ID** in the sample with the name of the connection for the remote LU.

Listing 3-20 Sample Remote Program Definition in BEASNA File

```
DEFINE PROGRAM(MIRRDPLS)    GROUP(BEASNA)
       DE(EAM EXAMPLE DPL REMOTE PROGRAM DEFINITION)
        LANGUAGE(C)         DATALOCATION(ANY)
        REMOTESYSTEM(**CONNECTION ID**) REMOTENAME(MIRRDPLS)
```

BEASNA File: Transaction Definition

The BEASNA file also contains a sample transaction definition, shown in Listing 3-21.

Listing 3-21 Sample Transaction Definitions in BEASNA File

```
DEFINE TRANSACTION(DTPS     GROUP(BEASNA)
       DE(EAM EXAMPLE CICS DTP SERVER)
       TASKDATALOC(ANY)     PROGRAM(TOUPDTPS)

DEFINE TRANSACTION(H1PL)    GROUP(BEASNA)
       DE(EAM EXAMPLE CICS DPL CLIENT - SYNCLEVEL 1)
       TASKDATALOC(ANY)     PROGRAM(MIRRDPLC)

DEFINE TRANSACTION(H2PL)    GROUP(BEASNA)
       DE(EAM EXAMPLE CICS DTP CLIENT - SYNCLEVEL 2)
       TASKDATALOC(ANY)     PROGRAM(MIRRDPLC)

DEFINE TRANSACTION(H0TP)    GROUP(BEASNA)
       DE(EAM EXAMPLE CICS DPL CLIENT - SYNCLEVEL 0)
       TASKDATALOC(ANY)     PROGRAM(MIRRDTPC)

DEFINE TRANSACTION(H2TP)    GROUP(BEASNA)
       DE(EAM EXAMPLE CICS DTP CLIENT - SYNCLEVEL 2)
       TASKDATALOC(ANY)     PROGRAM(MIRRDTPC)
```

BEASNA File: Partner Definition

The sample CICS/ESA client MIRRDTPC contains CPI-C verbs. The partner resource definition contains the CPI-C side information needed to allocate a conversation with the ATMI server and information about the remote LU and transaction program.

As shown in Listing 3-22, the `TPNAME` parameter identifies the transaction program that is invoked in the remote system. In this case, the name correlates to the `RNAME` value in the `DM_LOCAL_SERVICES` section of the `DMCONFIG` file. The `RNAME` there must match the `TPNAME` in the partner definition. (Notice in the sample `DMCONFIG` file that a local service definition `MIRROR` exists. The `RNAME` in that definition matches the `TPNAME` in the sample partner definition.)

The profile resource definition can define conversation attributes, in particular `MODENAME`. In the sample, the `PROFILE` parameter can be replaced with a valid profile resource definition. The default profile name for the parameter is `DFHCICSA`. `DFHCICSA` is a CICS-delivered profile.

Use the `NETNAME` specified in the Connection definition of the remote LU to replace `**NETWORK NAME**`.

Listing 3-22 Sample Partner Definition in BEASNA File

```
DEFINE PARTNER(MIRRDTPS)    GROUP(BEASNA)
DE(EAM EXAMPLE CICS DTP CLIENT USING CPIC VERBS
TPNAME(MIRRORSERV)          PROFILE(**DFHCICSA**)
NETNAME(**NETWORK NAME**)
```

View Connection and Session Status

Once you have made the verification group definitions, you can view the status of connections and sessions using the following CICS/ESA system commands:

```
CEMT I CONN(BEA)           **view the status of the connection
CEMT I NET(**NETNAME**)    **View the status of the sessions
CEMT I MODENAME(**MODE**)  **View the status of the mode
```

Running the Sample Application

Now you are ready to validate the eAM installation by running the sample application. You should have completed the following prerequisites:

- Install the eAM software
- Configure the ATMI platform and eAM local domains
- Configure the CICS/ESA remote domain
- Initialize the stack processes
- Start the servers

The sample application contains several scenarios. When the client runs as an ATMI client, the server runs as a CICS/ESA host. When the client runs as a CICS/ESA client, the server runs as an ATMI server.

Running the Application from an ATMI Client

In this scenario, the `toupclt` client performs a `tpcall` to the CICS/ESA host server. The server converts the text string you enter from lower-case to upper-case letters. The client may be run in transactional or non-transactional mode. The CICS/ESA server may be run as a DTP or DPL program

For example, enter the following command:

```
toupclt -s 0 -t DTP “hello world”
```

In this command, the arguments and options are defined in the following way:

`-s (0|2)`
: Application Sync-Level.
0 = none (default)
2 = Transactional.

`-t (DPL|DTP)`
: Remote Server Program.
DPL = Distributed Program Link (default)
DTP = Distributed Transaction Program

`-h`
: Help

“ “
: Lowercase text string of up to 1915 characters.

If the ATMI client successfully executes the command, it displays the text string in upper-case letters, for example:

```
“HELLO WORLD”
```

Running the Application from a CICS/ESA Client

The following sections depict two scenarios for running the application from a CICS/ESA client.

CICS/ESA Client with CPI-C

In this scenario, the CICS/ESA client sends a text string to the eAM simple server. The string is re-displayed on the client's screen in reverse order.

Enter either of the following commands:

```
H0TP <string>
```

```
H2TP <string>
```

In these commands, the arguments and options are defined in the following way:

```
H0TP = CPI-C Application Sync-Level 0
```

```
H2TP = CPI-C Application Sync-Level 2
```

```
<string> is a text string up to 1915 characters long.
```

If the CICS/ESA client successfully completes the transaction, it displays the text string in reverse order. For example:

You enter:

```
H0TP HELLO WORLD
```

The system returns:

```
DLROW OLLEH
```

CICS/ESA Client with DPL

Two transactions are available to execute the same program for the DPL sample. One is a simple request/response with the required sync-level 1, the other is a transactional request/response with sync-level 2.

Enter either of the following commands:

```
H1PL <string>
```

```
H2PL <string>
```

In these commands, the arguments and options are defined in the following way:

```
H1PL = DPL Application Sync-Level 1
```

```
H2PL = DPL Application Sync-Level 2
```

```
<string> is a text string up to 955 characters long.
```

If the CICS/ESA client successfully completes the transaction, it displays a reverse (or mirror) image of the text string concatenated to the input text string. For example:

You enter:

```
H1PL HELLO WORLD
```

The system returns:

```
HELLO WORLD : DLROW OLLEH
```

Running the Application from a CICS/VSE Client

The CICS/VSE sample applications are virtually identical to the CICS/ESA sample applications, except the file names have the `.vse` extension. The same scenarios for running the sample applications apply for both client categories. Other noteworthy differences are minor syntax changes in the sample applications to accommodate the CICS/VSE operating system and lack of CICS/VSE support for a CPI-C interface.

CICS/VSE Sample Applications

The following is a list of the CICS/VSE sample applications:

- `BEASNA.RDO.VSE`

- `MIRRDPLC.c.VSE`
- `TOUPDPLS.c.VSE`
- `TOUPDTPS.c.VSE`
- `TOUPDTPS.cbl.VSE`

Other Considerations

Some FTP operations may result in mistranslation of C-language special characters. It may be necessary for you to edit the source file(s) and correct the mistranslations of brackets [ ] and parallel bar | | characters.

CSMI and CVMI use profile `DFHCICST` with the attribute `INBFMH = NO`. When using DTP transactions, you are required to set `INBFMH = ALL`; otherwise, an `ABEND AXFQ` occurs.

CICS/VSE transactions should be defined with the security attribute `RSLC=NO`; otherwise an `EIBRCODE (NOTAUTH)` is returned to the client.

4 Security

Security refers to techniques for ensuring that data stored in a computer or passed between computers is not compromised. Most security measures involve passwords and data encryption. A password is a secret word or phrase that gives a user access to a particular program or system, and data encryption is the translation of data into a form that is unintelligible without a deciphering mechanism.

This section covers the following topics:

- Understanding eAM Security
- Mapping User IDs
- Using Encryption
- Using TCP/IP Link Authentication

Note: All references to ATMI files, functions, and documentation apply to Tuxedo, eLink Platform, and WebLogic Enterprise files, functions, and documentation.

Understanding eAM Security

Distributed applications such as those used for electronic commerce (e-commerce) offer many access points for malicious people to intercept data, disrupt operations, or generate fraudulent input. The more distributed a business becomes, the more vulnerable it is to attack. Thus, the distributed computing software or middleware, upon which such applications are built must provide security.

BEA eLink Adapter for Mainframe (eAM) works with the ATMI platform to enable the following security capabilities:

- User ID Mapping
- Encryption
- TCP/IP Link Authentication

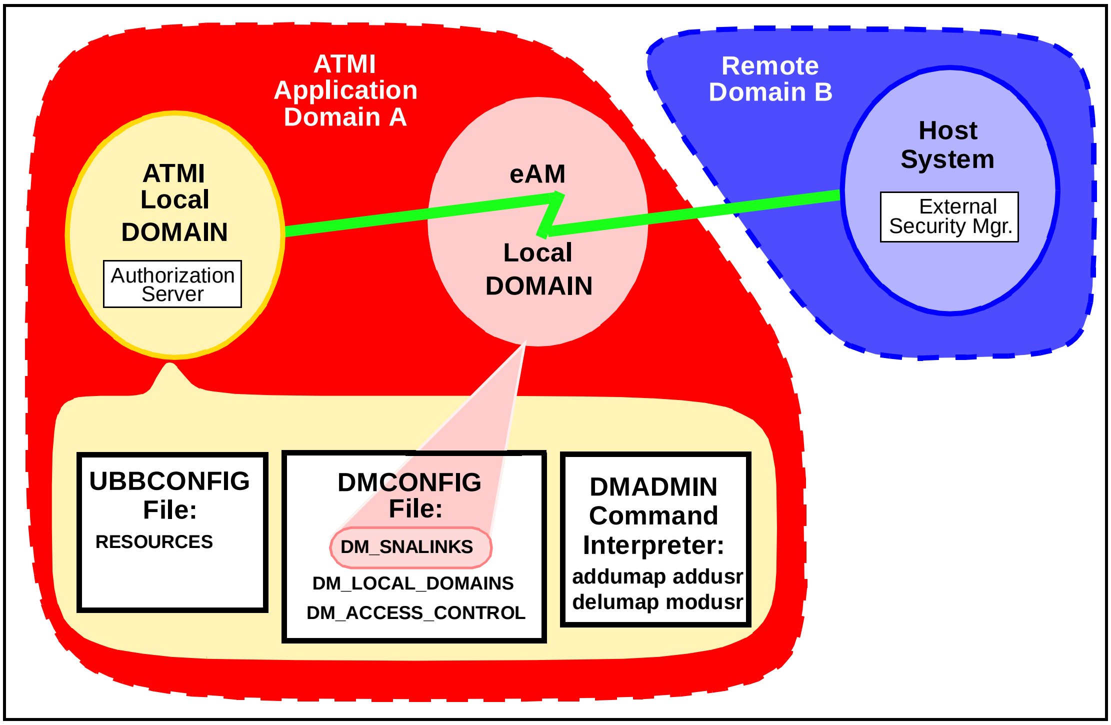
Mapping User IDs

User IDs are used to control access to system resources in the ATMI and mainframe environments. For user IDs to be used by those ATMI and mainframe environments, both sides must have security mechanisms in place. For the ATMI domain, the security mechanism is the Authorization Server. For the host system, the security mechanism is the External Security Manager. Figure 4-1 shows eLink Adapter for Mainframe (eAM) security elements.

The eAM software allows user IDs to be shared between domains in two ways:

- ATMI-to-Host User ID Mapping
- Direct User ID Mapping

Figure 4-1 eLink Adapter for Mainframe Security Elements



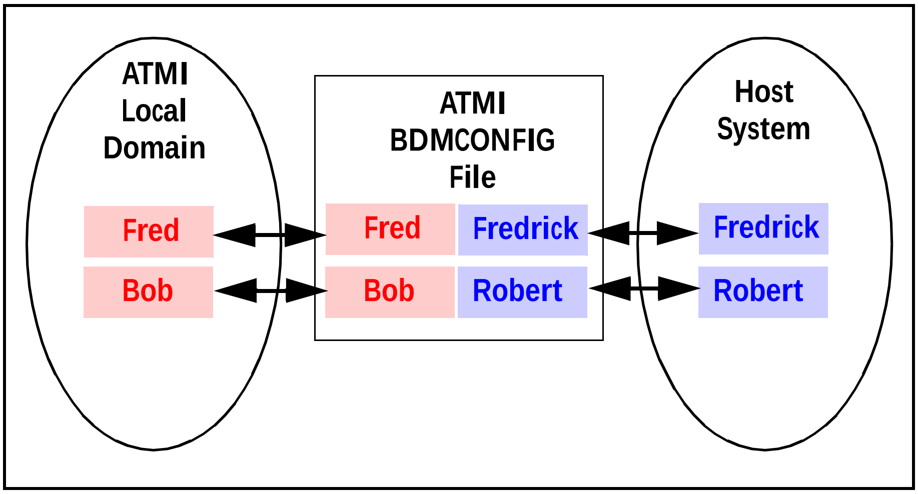
Caution: Mixed environment security is more complex than depicted in Figure 4-1. A domain without an operational security mechanism in place accepts all transaction requests by treating user IDs as "trusted users." Refer to the appropriate ATMI product documentation for more detailed information about domain security.

ATMI-to-Host User ID Mapping

ATMI-to-host user ID mapping associates a user ID in the local domain with a corresponding user ID in the host system. With ATMI-to-host user ID mapping, an ATMI user ID can be different from its counterpart on the host. See Figure 4-2.

The `dmadmin` commands are used to create this kind of mapping. Refer to the "Using dmadmin Commands to Administer User ID Mapping" section. These commands change the binary form of the `DMCONFIG` file (called the `BDMCONFIG` file).

Figure 4-2 Typical ATMI-to-host User ID Mapping



Direct User ID Mapping

Direct user ID mapping enables the direct mapping of an ATMI user ID to an identical host user ID. This eliminates the need to use the `dmadmin` commands, as with ATMI-to-host user ID mapping. When this feature is used, any user ID mappings in the `BDMCONFIG` file are bypassed. To enable this feature, specify a command-line parameter with the `GWSNAX` command when starting the eAM gateway. Refer to the "Bypassing User ID Mapping" section.

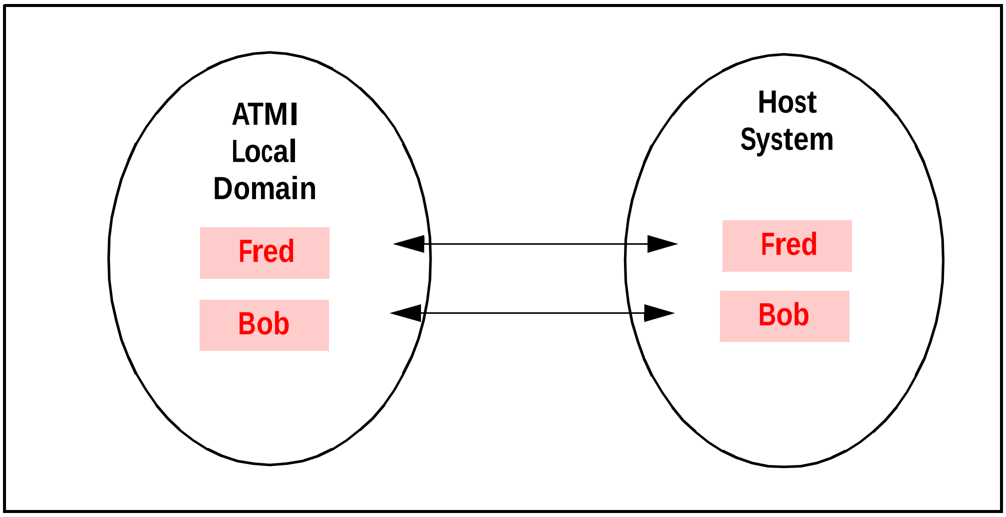
Note: When direct user ID mapping is used, modification or elimination of any BDMCONFIG file mapping entries is not necessary.

With direct user ID mapping, the user IDs in the ATMI and host environments must be identical as shown in Figure 4-3. When the ATMI local domain initiates a request, the ATMI user ID is applied to the requested host service. When the host initiates a request, the host user ID is applied to the requested ATMI service.

Notes: Identical user IDs must exist in the local domain and in the host domain for direct user ID mapping to be used.

With direct mapping, only security level IDENTIFY can be supported.

Figure 4-3 Direct User ID Mapping



Configuring User ID Mapping

Specify parameters bearing on local domain and eAM security in the DMCONFIG and UBBCONFIG files in the following four sections:

- DM_LOCAL_DOMAINS section of the DMCONFIG file
- DM_SNALINKS section of the DMCONFIG file

- DM_ACCESS_CONTROL section of the DMCONFIG file
- RESOURCES section of the UBBCONFIG file

Determining Security Parameters

The combined settings of the SECURITY parameters in the UBBCONFIG and the DMCONFIG files have the following effects:

- When the DM_LOCAL_DOMAINS security parameter is set to NONE or APP_PW, no action is taken by the gateway with regard to security.
- However, when the UBBCONFIG file security parameter is set to APP_PW, the application password is validated by an AUTHSVC when clients join the application. The AUTHSVC is provided by the user application.

If security is to be enforced by both the local domain and the host system for each request inbound from the host system to the local domain, the following settings must be made:

- The UBBCONFIG file SECURITY parameter must be set to one of USER_AUTH, ACL, or MANDATORY_ACL;
- The DMCONFIG file DM_LOCAL_DOMAINS section SECURITY parameter must be set to DM_USER_PW
- The DMCONFIG file DM_SNALINKS SECURITY parameter must be set to IDENTIFY or VERIFY.
- The SNA stack must be configured with the appropriate parameter for IDENTIFY or VERIFY.
- The ATTACHSEC level for the connection definition in the host system must be set to IDENTIFY or VERIFY to match the DMCONFIG file DM_SNALINKS SECURITY parameter.

Determining Security Parameters for Inbound Requests

Table 4-1 shows settings for the SECURITY parameters in the UBBCONFIG and DMCONFIG files required to achieve local domain and host system security combinations for inbound requests from the host system.

Note: Security setting combinations other than those shown in the tables will have unpredictable results.

Table 4-1 Security Settings for Inbound Requests from Host Systems

Security Combinations		**Settings**			
Local	**Host**	**`UBBCONFIG SECURITY`**	**`DM_LOCAL_DOMAINS SECURITY`**	**`DM_SNALINKS SECURITY`**	**Remote Verification**
No	No	`NONE or APP_PW`	`NONE or APP_PW`	`LOCAL`	Not Applicable
Yes	No	`USER_AUTH, ACL, or MANDATORY_ACL`	`DM_USER_PW`	`LOCAL`	INVALID
No	Yes	`NONE or APP_PW`	`NONE or APP_PW`	`IDENTIFY or VERIFY`	Not Applicable
Yes	Yes	`USER_AUTH, ACL, or MANDATORY_ACL`	`DM_USER_PW`	`IDENTIFY or VERIFY`	UID or UID+PW

For requests sent from the host system, the local domain extracts the remote user ID, or user ID and password, from the conversation start-up request and checks the domain security table. That table contains pairs of local principal user IDs and remote user IDs, maintained on a service-by-service basis. The remote user ID is mapped to the local principal user ID. The local principal user ID and password are used for further ACL checking, as specified in the `UBBCONFIG` file. If the direct user ID mapping option is specified, the remote user ID is used as the local principal user ID.

When a request is received from the host system, the local domain checks the ACL in the `DMCONFIG` file for the local service to see if requests from the remote domain are permitted. If the `DMCONFIG` file does not contain an ACL for the local service, the service is accessible to all requests.

Determining Security Parameters for Outbound Requests

If security is to be enforced by both the local domain and the host system for each request outbound from the local domain, the following settings must be made:

- The UBBCONFIG file SECURITY parameter must be set to one of USER_AUTH, ACL, or MANDATORY_ACL
- The DMCONFIG file DM_LOCAL_DOMAINS section SECURITY parameter must be set to DM_USER_PW
- The DMCONFIG file DM_SNALINKS SECURITY parameter must be set to IDENTIFY or VERIFY
- The SNA stack must be configured with the appropriate parameter for IDENTIFY or VERIFY.
- The ATTACHSEC level for the connection definition in the host system must be set to IDENTIFY or VERIFY to match the DMCONFIG file DM_SNALINKS SECURITY parameter.

Table 4-2 shows settings for the SECURITY parameters in the UBBCONFIG and DMCONFIG files required to achieve local domain and host system security combinations for outbound requests.

Note: Security setting combinations other than those shown in the tables will have unpredictable results.

Table 4-2 Security Settings for Outbound Requests from Local Domain

Security Combinations		**Settings**			
Local	**Host**	**`UBBCONFIG SECURITY`**	**`DM_LOCAL_DOMAINS SECURITY`**	**`DM_SNALINKS SECURITY`**	**Remote Verification**
No	No	`NONE or APP_PW`	`NONE or APP_PW`	`LOCAL`	Not Applicable
Yes	No	`USER_AUTH, ACL, or MANDATORY_ACL`	`DM_USER_PW`	`LOCAL`	Not Applicable
No	Yes	`NONE or APP_PW`	`NONE or APP_PW`	`IDENTIFY or VERIFY`	INVALID
Yes	Yes	`USER_AUTH, ACL, or MANDATORY_ACL`	`DM_USER_PW`	`IDENTIFY or VERIFY`	UID or UID+PW

For a request sent to the host system, the local principal user ID is located in the domain security table and the associated remote user ID, or user ID and password, are put into the conversation start-up request before being sent over the LU6.2 conversation. This situation occurs if `SECURITY` is set to `IDENTIFY` or `VERIFY` in the `DM_SNALINKS` section of the `DMCONFIG` file. If the direct user ID mapping option is specified, the local principal user ID is put into the conversation startup request.

Setting DMCONFIG File Security Parameters

Three sections in the `DMCONFIG` file contain parameters affecting eAM control of access to the ATMI local domain:

- `DM_LOCAL_DOMAINS` section contains a `SECURITY` parameter which specifies the type of security enforced for the ATMI local domain.
- `DM_SNALINKS` section contains a `SECURITY` parameter that records the security in effect for the host system.

- DM_ACCESS_CONTROL section contains local access control lists used by the ATMI local domain to associate local resources with host systems permitted to have access to them.

Caution: Do not delete the DMCONFIG binary file before running the dmloadcf command. Tables of remote users, remote passwords, and remote mappings are stored in this file. If deleted, all security information must be re-entered.

DM_LOCAL_DOMAINS Section

The SECURITY parameter settings in this section work in conjunction with the SECURITY parameter in the RESOURCES section of the ATMI local domain's UBBCONFIG file to establish how eAM controls access to the ATMI local domain. The parameter takes the form:

```
SECURITY = {value}
```

In this parameter, value can be set as:

NONE
: No security is enforced.

APP_PW
: No security is enforced.

DM_USER_PW
: User and password security is enforced.

If this parameter is set to NONE or APP_PW, the local domain takes no action with regard to security. If this parameter is set to DM_USR_PW, the local domain enforces security according to the setting in the UBBCONFIG file (refer to "Setting DMCONFIG File Security Parameters").

DM_SNALINKS Section

This section of the DMCONFIG file is dedicated to eAM parameters. It records the security in effect for the host system. It correlates to the values set for the ATTACHSEC parameter in the connection resource definition. In the following parameter definition, *remote* refers to the ATMI domain and *local* refers to the host system. The parameter takes the form:

```
SECURITY_TYPE = {value}
```

In this parameter, `value` can be set as:

`LOCAL`
: Specifies that a user identifier is not to be supplied by the remote system. `LOCAL` is the default value.

`IDENTIFY`
: Specifies that a user identifier is expected on every attach request. All remote users of a system must be identified to the remote external security manager.

`VERIFY`
: Attaches a user ID and valid password to the remote region. The user ID and password are controlled by the region's external security manager.

`PERSISTENT`
: Not fully supported. Functions the same as `VERIFY`.

`MIXIDPE`
: Not fully supported. Functions the same as `VERIFY`.

The values `LOCAL` and `IDENTIFY` are roughly equivalent to `NONE` and `APP_PW` for the `SECURITY` parameter in the `DMCONFIG` file.

DM_ACCESS_CONTROL Section

This section contains local ACL used by the ATMI local domain to restrict access by remote regions to local resources. Refer to the "Security Administration" section in the *BEA ATMI Administrator's Guide*.) Each entry consists of an `ACL_NAME` resource identifier along with a list of required parameters designating remote domains permitted to access the resource. If no entry exists for a local service, the service is accessible to all remote domains.

Setting UBBCONFIG File Security Parameters

The `RESOURCES` section in this file contains a `SECURITY` parameter that works in conjunction with the `SECURITY` parameter in the `DMCONFIG` file to establish how eAM controls access to the ATMI local domain. This parameter takes the form:

```
SECURITY = {value}
```

In this parameter, `value` can be set as:

`NONE`
: No security is enforced (default).

`APP_PW`
: Requires password authorization for the gateway and administrative tools to connect to the local application.

`USER_AUTH`
: Same as `APP_PW`, but additional authorization is required on a per-user basis.

`ACL`
: Same as `USER_AUTH`, but additional access-control checks are done on service names, queue names, and event names. If no *Access Control Lists (ACL)* exists for a given name, access is granted.

`MANDATORY_ACL`
: Same as `ACL`, but if no ACL exists for a given name, access is denied.

In most cases, the `UBBCONFIG` file has already been configured and you do not need to establish the `SECURITY` parameter settings, but examining this file enables you to see how eAM enforces security.

If this parameter is set to `NONE`, no security is enforced. If set to `APP_PW`, the local ATMI domain's Authorization Server prompts for the application password. If set to `USER_AUTH`, `ACL`, or `MANDATORY_ACL`, the qualified security is enforced as specified.

Bypassing User ID Mapping

To use direct user ID mapping, use the `-m` parameter in the `GWSNAX` process start-up command line entry. This parameter allows you to establish direct user ID mapping, rather than ATMI-to-host user ID mapping.

Note: If you bypass user ID mapping, the local and host domains must have identical user IDs in effect, otherwise a security error occurs.

For example, to set the gateway server process to bypass user ID mapping, enter a command in the following format:

```
GWSNAX SRVGRP = <groupname> SRVID = <number>  CLOPT = "-A -- -m"
```

Refer to `GWSNAX` in Appendix A, "Reference Pages" for more information.

Using dmadmin Commands to Administer User ID Mapping

When ATMI-to-host user ID mapping is used, you must create mappings in the `BDMCONFIG` file.

Note: If the direct user ID mapping option is specified, creation of mappings in the `BDMCONFIG` file is not necessary. Any mappings in the `BDMCONFIG` file are ignored.

User ID mapping between the local domain and the host system is configured using the `addumap`, `addusr`, `delumap`, `modusr`, and `delusr` commands of the `dmadmin` utility to accomplish the following tasks:

- Adding a User ID and Password
- Mapping a User ID
- Removing User ID Mapping
- Deleting a User ID and Password
- Modifying a Password

Refer to Appendix A, "Reference Pages" for more information about each ATMI command.

To use these commands, enter `dmadmin` at the system prompt. At the `dmadmin` ">" prompt, enter the commands as described.

Adding a User ID and Password

Use the `addusr` ATMI command to add an ATMI local domain user ID and password to the remote domain user and password file. Enter the following command:

```
addusr {-d} local_domain_id {-R} remote_domain_id {-u}
remote_userid [{-w}]
```

The arguments and options in this command are defined in the following way:

`-d`

Specifies the name of the local domain with which the user ID and password are associated.

`-R`
: Specifies the name of the remote domain to which the user ID and password are added.

`-u`
: Specifies the user name to be added. Enter the user's password when prompted.

`-w`
: Specifies not to prompt for password. Use when running with `IDENTIFY`.

Mapping a User ID

Use the `addumap` ATMI command to map a local domain principal user ID number to a remote domain user name. The user ID must be added before it can be mapped. Refer to the "Adding a User ID and Password" section. Enter the following command:

```
addumap {-d} local_domain_id {-R} remote_domain_id
{-p} local_principal_userid {-u} remote_userid
```

The arguments and options in this command are defined in the following way:

`-d`
: Specifies the name of the local domain with which the user ID is associated.

`-R`
: Specifies the name of the remote domain to which the user ID is mapped.

`-p`
: Specifies the local principal user ID number defined in the `tpusr`.

`-u`
: Specifies the remote user name as defined in the security application of the remote domain.

Removing User ID Mapping

Use the `delumap` ATMI command to remove the mapping for a local domain principal user ID to a remote domain user name. Enter the following command:

```
delumap {-d} local_domain_id {-R} remote_domain_id
{-p} local_principal_userid {-u} remote_userid
```

The arguments and options in this command are defined in the following way:

`-d`
: Specifies the name of the local domain with which the user ID is associated.

`-R`
: Specifies the name of the remote domain to which the user ID is mapped.

`-p`
: Specifies the local principal user ID number defined in the `tpusr`.

`-u`
: Specifies the remote user name as defined in the security application of the remote domain.

Deleting a User ID and Password

Use the `delusr` ATMI command to remove a local ATMI domain user ID and password from the remote domain user and password file. The mapping for a user ID must be removed before the user ID can be removed. Enter the following command:

```
delusr {-d} local_domain_id {-R} remote_domain_id {-u}
remote_userid
```

The arguments and options in this command are defined in the following way:

`-d`
: Specifies the name of the local domain with which the user ID and password are associated.

`-R`
: Specifies the name of the remote domain from which the user ID and password are to be deleted.

`-u`
: Specifies the user name to be deleted.

Modifying a Password

Use the `modusr` ATMI command to modify a local domain user’s password recorded in a remote domain’s user and password file. Enter the following command:

```
modusr {-d} local_domain_id {-R} remote_domain_id {-u}
remote_userid [{-w}]
```

The arguments and options in this command are defined in the following way:

`-d`
: Specifies the name of the local domain the user ID and password are associated with.

`-R`
: Specifies the name of the remote domain in which the user ID and password are to be modified.

`-u`
: Specifies the user name to be modified. Enter the user's password when prompted.

`-w`
: Specifies not to prompt for password. Use when running with `IDENTIFY`.

Setting Security Scenario

This section provides an example of step-by-step instructions for setting up security in an application that has already been configured.

Configuring Security in the ATMI Domain

1. Edit the `UBBCONFIG` file.
   a. In the `RESOURCES` section, add `SECURITY USER_AUTH`.
   b. In the `SERVERS` section, add the `AUTHSVR` server.

   Note: `SECURITY USER_AUTH` level implies that application passwords, user IDs, and user passwords are required to join the application. `AUTHSVR` is the ATMI-supplied authentication server. It advertises the service `AUTHSVC`.

2. Enter the `tmloadcf` command to load the ATMI configuration, for example:

   ```
   tmloadcf -y ubbconfig.sna
   ```

3. Set the application password. (The `tmloadcf` command prompts for the application password.)

4. Add users to the ATMI domain by using the `tpusradd` command. The command prompts for each password, for example:

   ```
   tpusradd me
   ```

   (Enter password for `me`.)

   Note: Do not use the command `tpaddusr`.

5. Modify the ATMI client to specify security parameters in the `tpinit` call. Listing 4-1 is an example of the code to do this.

Listing 4-1 Security Parameters Added to `tpinit` Call

```
    TPINIT *tpinitbuf;
    char passwd[30];
    int security_level;

/* Initialize security parameters */

    if ((tpinitbuf = (TPINIT *) tpalloc("TPINIT", NULL,
    TPINITNEED(sizeof(passwd)))) == NULL)

    {
       userlog("tpalloc tpinit failed %s \n", tpstrerror(tperrno));
       exit(1);
    }
    strcpy(tpinitbuf->usrname,"");
    strcpy(tpinitbuf->cltname,"");
    strcpy(tpinitbuf->passwd,"");
    strcpy(tpinitbuf->grpname,"");


/* Determine level of enforced security */
    security_level = tpchkauth();

    if ((security_level == TPSYSAUTH) || (security_level ==
    TPAPPAUTH))

    {
       fprintf(stdout,"\nApplication passwd required.");
       fprintf(stdout,"\nApplication passwd:");
       gets(tpinitbuf->passwd);
    }

    if (security_level == TPAPPAUTH)

    {
       fprintf(stdout,"\nUser Name required.");
       fprintf(stdout,"\nUser Name:");
       gets(tpinitbuf->usrname);

       fprintf(stdout,"\nUser Password required.");
       fprintf(stdout,"\nUser Password:");
       gets(passwd);
       strcpy(&tpinitbuf->data,passwd);
       tpinitbuf->datalen=strlen(passwd);
    }

    if (tpinit(tpinitbuf) == -1)
    {
        userlog("TPINIT %s \n", tpstrerror(tperrno));
        exit(1);
    }
```

6. Verify security in the ATMI domain by running the client.
7. Enter the `dmloadcf` command to load the domain configuration. For example:

   ```
   dmloadcf -y dmconfig.sna
   ```
8. Enter the `tmboot` command to boot the ATMI domain, for example:

   ```
   tmboot -y
   ```
9. Configure security for the SNA domain by editing the `DMCONFIG` file.
   a. In the `DM_LOCAL_DOMAINS` section, add the parameter:

      ```
      SECURITY=DM_USER_PW
      ```
   b. In the `DM_SNALINKS` section, add the parameter for the remote link:

      ```
      SECURITY=VERIFY
      ```
10. Add the user name mapping for the remote domain by invoking `dmadmin` and using the `addumap` command to map local user IDs to remote user IDs. For example:

    ```
    dmadmin
    >addumap -d myldom -R myrdom -p localme -u REMOTEME
    ```
11. Add a password for remote user IDs for the remote domain by invoking `dmadmin` and using the `addusr` command to provide remote password(s). For example:

    ```
    dmadmin
    >addusr -d myldom -R myrdom -u REMOTEME
    ```

    (The system responds with the following prompts:

    ```
       ERROR: Enter Remote User’s Password:
       ERROR: Re-enter Remote User’s Password:)
    ```

Configuring Security in the Local Domain

To configure security in the local domain:

- Set the security parameters in the SNA stack.

Refer to the appropriate stack documentation.

Configuring Security in the Remote Domain

Change the security of connection definitions on the mainframe host by doing the following:

1. Expand the group that contains the connection definitions. For example:

   ```
   CEDA EX GR(MYCONNGRP)
   ```

   Replace `MYCONNGRP` with the name of the group that contains your connection definitions.

2. Alter security of each connection definition by changing the value of `ATTACHSEC` to `VERIFY` on each connection definition.

3. Put each connection out of service by inquiring on the connection's `CEMT I CONN(MYCN)` and tabbing to the connection, then changing the `INS` entry to `OUT`.

4. Install the modified connection definitions. For example:

   ```
   CEDA I GR(MYCONNGRP)
   ```

   Replace `MYCONNGRP` with the name of the group that contains your connection definitions.

5. The security definition is complete. Run the application.

Setting the Security Level to IDENTIFY

To configure the previous example with a security level of `IDENTIFY`, complete the following steps:

1. Change the `SECURITY` parameter in the `DMCONFIG` file to `IDENTIFY`.

2. Change the `ATTACHSEC` parameter on the connection to `IDENTIFY`.

3. Change the remote user password by using the `addusr -w` option so that no password is specified as in the following example:

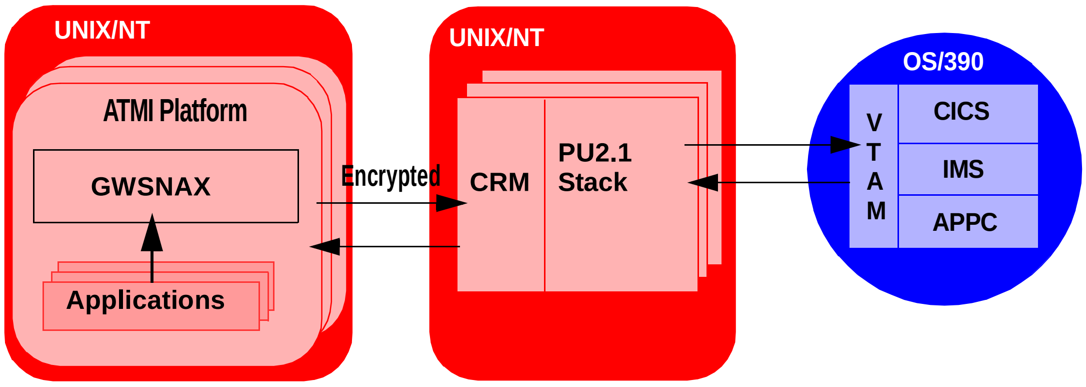
```
addusr -d myldom -R myrdom -u REMOTEME -w
```

Using Encryption

To establish secure communications between the Communication Resource Manager (CRM) and the gateway (GWSNAX) over a distributed network, eAM uses a link-level encryption process. As illustrated in the following diagram, this encryption feature only applies to the link between the eAM gateway and the CRM.

Note: The appropriate ATMI security add-on (40 bit or 128 bit) must be purchased to enable encryption.

Figure 4-4 Encrypted Links



The encryptions process occurs in the following way:

1. When the gateway establishes a connection to the CRM, the entities exchange messages to determine if encryption is enabled.
2. If both entities have encryption capability, a negotiation is performed to determine the level of encryption established.

   Each process has a range of acceptable encryption levels, as specified on the process start-up command line. The lowest common level of encryption is used.

Note: When encryption is established for communications between the CRM and the gateway, system performance may deteriorate. The higher the encryption level, the more likely deterioration may occur.

Configuring the eAM Gateway and CRM for Encryption

To configure the eAM gateway:

1. Determine acceptable range of encryption levels (`min` and `max`).
2. Edit the `GWSNAX` entry in the `UBBCONFIG` file to add the `-n` option with the desired `min` and `max`.

   See `GWSNAX` in Appendix A, "Reference Pages."

To configure the CRM:

1. Determine acceptable range of encryption levels (`min` and `max`).
2. Configure the CRM in one of the following ways:
   - If the CRM is started from the command line, add the -n option with the desired `min` and `max`, as described in `SNACRM` in Appendix A, "Reference Pages."
   - If the CRM is started as an ATMI server, modify the `SNACRM` server entry to contain the `-n` option with the desired `min` and `max`, as described in `SNACRM` in Appendix A, "Reference Pages."

If `crmlkoff`, `crmlkon`, or `crmdown` are used with encrypted CRM, no additional command line arguments are needed.

Using TCP/IP Link Authentication

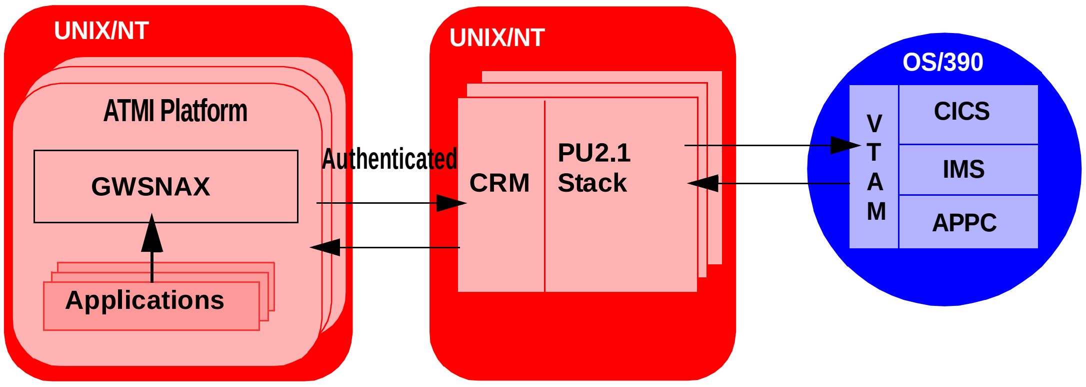
In addition to encryption, eAM uses an authentication process to establish secure communications between the CRM and gateway over a distributed network. Table 4-3 lists the processes that support authentication.

Table 4-3 CRM Processes Supporting Authentication

Process	Encryption Capability
crmlkon	Supports one-way authentication; CRM authenticates the `crmlkon` program, not vice versa
crmlkoff	Supports one-way authentication; CRM authenticates the `crmlkoff` program, not vice versa
crmdown	Supports one-way authentication; CRM authenticates the `crmdown` program, not vice versa
GWSNAX	Supports two-way authentication
CRM	Supports two-way authentication

As illustrated in the following diagram, this authentication feature only applies to the link between the gateway and the CRM.

Figure 4-5 Authenticated Links



When the gateway establishes a connection to the CRM, the following events occur:

1. Each entity issues a challenge.
   - The challenge is based on a random number combined with an authentication key.

- The authentication key is contained in a key file designated by the process command line specification.

2. Each entity issues a response to the challenge it receives. This response is based on the challenge combined with the entity's authentication key.
3. Each entity verifies the response by comparing the response to its own calculated result.
   - If the challenge/response exchange fails, the connection is closed and an error is logged.
   - If the challenge/response succeeds, full communications are enabled.

Configuring the eAM Gateway and CRM for Authentication

To configure the gateway for authentication, complete the following steps:

1. Establish an authentication key file.
   - Create a text file containing the authentication key. This key should be no more than eight characters. Communicating processes must have the same entry in their key files for authentication to be successful.
   - Store the `keyfile` in a protected location.
2. Use a general command line entry with the following format to establish authentication, as described in `GWSNAX` in Appendix A, "Reference Pages":

```
[-u <keyfile>]
```

To configure the CRM:

1. Establish an authentication key file.
   - Create a text file containing the authentication key. This key should be no more than eight characters. Communicating processes must have the same entry in their key files for authentication to be successful.
   - Store the `keyfile` in a protected location.

2. Configure the CRM in one of the following ways:

   - If the CRM is started from the command line, add the `-u<keyfile>` option, as described in `SNACRM` in Appendix A, "Reference Pages."

   - If the CRM is started as an ATMI server, modify the `SNACRM` server entry to contain the `-u<keyfile>` option as described in `SNACRM` in Appendix A, "Reference Pages."

3. If `crmlkoff`, `crmlkon`, or `crmdown` are used with a CRM with authentication enabled, use the `-u<keyfile>` command line option as described in `SNACRM` in Appendix A, "Reference Pages."

5 Data Translations

Due to differences in platforms, operating systems, and programming languages, the data format used by applications in the ATMI domain differs significantly from that expected by applications in the remote domain. In the ATMI domain, applications are typically written in the C language, using C structures and ATMI typed buffers, such as VIEW and FML. In the remote domain, the C and COBOL programming languages are commonly used, with C structures or COBOL copybook definitions. The translation between these data types for both inbound and outbound messages occurs in the eAM gateway. The gateway has a number of configuration options for dealing with data conversion between domain types.

This section discusses the following data translation topics:

- Data Conversion
- Translation Rules for VIEW Data Types
- Using Information Integrator

Note: All references to ATMI files, functions, and documentation apply to Tuxedo, eLink Platform, and WebLogic Enterprise files, functions, and documentation.

Data Conversion

Like other domain gateways, the eAM gateway (GWSNAX) uses ATMI typed buffers to transmit and receive data. Since the remote host application does not understand the typed buffer, the ATMI application must communicate with the host application by using an aggregate data type known as a *record*. A *record* is a flat data area defined by a template that describes the data type and length of each field in the record.

The application developer should determine the format and content of the record structure used on the remote host, as well as the ATMI typed buffer to be used, before configuring the gateway.

In most cases the conversion between ATMI typed buffers and record formats is handled by the eAM gateway. The service definitions in the `DM_LOCAL_SERVICES` and the `DM_REMOTE_SERVICES` section of the `DMCONFIG` file provide parameters to describe the typed buffer/record combination required for successful communications between the applications.

Note: The current size of remote host messages is limited to approximately 32K bytes. Any conversions resulting in records larger than 32756 bytes are not supported.

Conversion of ATMI Typed Buffers to Records

When an ATMI application sends a typed buffer to a remote host application, the buffer must be converted to a record by the eAM gateway before it is passed to the remote host application. The gateway uses the service definition to determine what, if any, conversion must be applied to the buffer. The service definition uses the `INBUFTYPE` in both the `DM_LOCAL_SERVICES` and `DM_REMOTE_SERVICES` section of the `DMCONFIG` file to describe the desired conversion.

`INBUFTYPE` is specified in the following way:

```
INBUFTYPE = type:subtype
```

In this parameter definition, `type` must be one of the designated ATMI typed buffers described in the following subsections.

The `subtype` value names a view and is required for certain ATMI typed buffers.

Only one `type:subtype` may be entered for the `INBUFTYPE` parameter.

The following sections describe the conversions performed for each ATMI typed buffer.

Data Conversion for STRING Typed Buffer

By default, a null-terminated string is converted to `EBCDIC`. The null character is part of the converted record. See the "Translation Rules for Strings" section for more conversion options.

Data Conversion for X_OCTET/CARRAY Typed Buffers

No data conversion is performed on these typed buffers. The ATMI application or remote host application performs all conversion of data fields in the record, including all numeric and EBCDIC conversions.

These typed buffers are used when a data record cannot be described or converted using one of the other *strong* typed buffers. *Strong* means that eAM gateway can understand all data fields and perform the required data conversions.

These typed buffers are also options when the remote service expects many styles of data aggregation (multiple record types), because the `INBUFTYPE` parameter is limited to one `type:subtype`.

Data Conversion for VIEW/VIEW32/X_C_TYPE/X_COMMON Typed Buffers

A subtype is required for these typed buffers. The subtype is the name of the view that describes the remote host record. The ATMI buffer is converted from a C structure to the record, including EBCDIC conversion, using the compiled VIEW file. By default, the record is a COBOL structure, mapped by the remote host application using a COBOL copybook. See "Translation Rules for VIEW Data Types" for more conversion options.

Data Conversion for FML/FML32 Typed Buffers

A subtype is required for these typed buffers. The subtype is the name of the view that describes the remote host record. The data in the typed buffer is Field Manipulation Language (FML) data. The eAM gateway converts the buffer to a record described by the view, including EBCDIC conversion.

The ATMI buffer is converted from an FML typed buffer to a C structure using the subtype compiled VIEW file. The C structure is then converted to the record using the same subtype compiled VIEW file. By default, the record is a COBOL structure that is mapped by the remote host application using a COBOL copybook.

Conversion of Records to ATMI Typed Buffers

When a remote application sends a record to an ATMI application, the record must be converted to an ATMI typed buffer by the eAM gateway before it is passed to the ATMI application. The gateway uses the service definition to determine what, if any, conversion must be applied to the host record. The service definition uses the `OUTBUFTYPE` in both the `DM_LOCAL_SERVICES` and `DM_REMOTE_SERVICES` section of the `DMCONFIG` file to describe the desired conversion.

`OUTBUFTYPE` is specified in the following way:

```
OUTBUFTYPE=type:subtype
```

In this parameter definition, `type` must be one of the designated ATMI typed buffers described in the following subsections. The `type` not only determines the typed buffer, but also describes the host record.

The `subtype` value names a view and is required for certain ATMI typed buffers.

Only one `type:subtype` may be entered for the `OUTBUFTYPE` parameter.

The following sections describe the conversions performed for each ATMI typed buffer.

Data Conversion for STRING Typed Buffer

The null terminated string is converted to ASCII. The converted string is moved to an ATMI STRING typed buffer. See the “Translation Rules for Strings” section for more conversion options.

Data Conversion for X_OCTET/CARRAY Typed Buffers

No data conversion is performed on these typed buffers. The remote host application or the ATMI application converts the data fields in the record, including all numeric and ASCII conversions.

These typed buffers are used when the data record cannot be described or converted using one of the other *strong* type buffers. *Strong* means eAM can understand all data fields and perform the required data conversion.

These typed buffers are also options when the remote service expects many styles of data aggregation (multiple record types), because the `OUTBUFTYPE` parameter is limited to one `type:subtype`.

Data Conversion for VIEW/VIEW32/X_C_TYPE/X_COMMON Typed Buffers

A subtype is required for these typed buffers. The subtype is the name of the view that describes the remote host record. The remote host record is converted to an ATMI typed buffer. The compiled VIEW file is used to perform the data and ASCII conversion on the host record. By default, the host record is a COBOL data aggregate and the converted typed buffer is mapped by the ATMI application using a C structure. See the "Translation Rules for VIEW Data Types" section for more conversion options.

Data Conversion for FML/FML32 Typed Buffers

A subtype is required for these typed buffers. The subtype is the name of the view that describes the remote host record. The host record is converted to an FML buffer that is passed to the ATMI application.

By default, the host record is a COBOL record aggregate data type. The data is converted to a C structure, including ASCII conversion, using the compiled VIEW file. This data is then converted to an FML buffer using the field definitions associated with the VIEW.

Data Conversion For DPL Services

The eAM system supports remote CICS services as Distributed Program Link (DPL) programs. The DPL support is performed as if the ATMI service is a peer CICS/ESA service.

In a DPL program, the application is protected from all Distributed Transaction Processing (DTP). The DPL application is a request/response service, with all data communication performed by the passing of a `COMMAREA`.

A basic DPL request API looks like this:

```
EXEC CICS LINK

       PROGRAM()
       DATALENGTH()
       LENGTH()
       COMMAREA()
```

In the preceding example, the requester sends the `COMMAREA` of `DATALENGTH` size and the receiving application receives the `COMMAREA` data contents in a buffer the size of `LENGTH`. The `DATALENGTH` size might be smaller than the `LENGTH` size, but the requester expects and receives the response in the original `COMMAREA` buffer the size of `LENGTH`.

The difference between a DPL program and an ATMI service is that a receiving ATMI service can resize a reply buffer, while the DPL program expects a reply buffer of a designated size. Also, an ATMI requester can receive a resized buffer in a buffer different from the original reply buffer.

The eAM software performs the manipulation described in the following subsections to smoothly adjust to the requirements of both types of applications.

DPL Requests Originating From An ATMI Application

The eAM software must determine what size `COMMAREA` the remote DPL service is expecting because no corresponding `LENGTH` parameter exists on an ATMI request.

To determine the `LENGTH` value for a DPL request, the software uses the larger potential size of the `INBUFTYPE` or the `OUTBUFTYPE` parameter definitions, as described in Table 5-1.

The remote DPL application receives a buffer of `LENGTH` size and returns a buffer of `LENGTH` size.

Table 5-1 DPL Request LENGTH Calculation

INBUFTYPE or OUTBUFTYPE	LENGTH CALCULATION
STRING/X_OCTET/ CARRAY	For these typed buffers, only the `INBUFTYPE` parameter definition is used to determine the `LENGTH`.
VIEW/VIEW32/ X_COMMON/ X_C_TYPE	`LENGTH` is the maximum size of the VIEW file. This calculation takes in the potential size of both the C structure and the target record.
FML/FML32	The maximum size of the VIEW file. This calculation takes in the potential size of both the C structure, and the target record. The length of the FML buffer is not taken into account.

If no `LENGTH` can be determined, then the largest value allowed by the CICS application is allocated. Refer to the "Using Information Integrator" section.

DPL Requests Originating From a CICS DPL

The eAM software receives a `LENGTH` value and `COMMAREA` data of `DATALENGTH` size from the requesting CICS DPL. The software allocates a buffer of `LENGTH` size and moves the `COMMAREA` data into this buffer before performing the conversions.

Translation Rules for VIEW Data Types

The following sections on default data translation rules provide suggestions to help you:

- Develop VIEW definitions for input and output buffers and records
- Understand how string data and numeric data are treated with the VIEW data type

Table 5-2 lists VIEW data translation rules.

Table 5-2 VIEW Data Translation Rules

Field Type	Translation Rules
CARRAY	Passed without translation as sequences of bytes.
STRING and CHAR	Translated from ASCII to EBCDIC (If needed. Refer to "Translation Rules for Strings.")
SHORT	Translated to S9(4)COMP
LONG	Translated to S9(9)COMP
FLOAT	Translated to COMP-1
DOUBLE	Translated to COMP-2

Note: The ATMI platform provides a field type named `dec_t` that supports decimal values within VIEWs. The gateway translates these fields into machine independent representations of packed decimals. For example, `dec_t(m,n)` becomes `S9(2*m-(n+1))V9(n) COMP-3`. Therefore, a decimal field with a size of `8,5` corresponds to `S9(10)V9(5) COMP-3`.

The translation rules between C and IBM/370 data types are listed in Table 5-3.

Table 5-3 Data Translation Rules between C and IBM/370 Data Types

Remote Data Type	Description	View Field Type/Length
PIC X(n)	Alpha-numeric Characters	string / n
PIC X	Single Alpha-numeric Character	char
PIC X(n)	Raw Bytes	carray / n
PIC X	Single Numeric Byte	carray / 1
PIC S9(4) COMP	32-bit Integer	short

Remote Data Type	Description	View Field Type/Length
PIC S9(9) COMP	64-bit Integer	long
COMP-1	Single-precision Floating Point	float
COMP-2	Double-precision Floating Point	double
PIC S9((m+(n+1))/2)V9(n) COMP-3	Packed Decimal	dec_t / m,n

String Considerations

When you create VIEW definitions for input and output records that are used by CICS/ESA applications, do not specify an extra position for the terminating null characters that are used in string fields. For example, when a CICS/ESA application program expects 10 characters in an input record, specify 10 for that field, not 10 plus 1.

Note: Although eAM software does not require strings to be null-terminated, it respects null termination. When the gateway software detects a null (zero) character within a string, it does not process any subsequent characters. To pass full 8-bit data that contains embedded null values, use a CARRAY type field or buffer.

The character set translations performed by eAM are fully localizable, in accordance with the X/Open XPG Portability Guides. ASCII and EBCDIC translations are loaded from message files. Refer to the “Translation Rules for Strings” section for more information.

The eAM software contains default behaviors that should meet the requirements of most English-language applications. However, you may find it necessary to customize the translation. Refer to the “Translation Rules for Strings” section for more information.

Converting Numeric Data

You can convert numeric data into different data types easily, if you specify enough range in the intermediate and destination types to handle the maximum value needed.

For example, you can convert a Field Manipulation Language (FML) field of double into a packed decimal field on the remote target system by specifying an appropriate `dec_t type` VIEW element.

You can also convert numeric values into strings. For example, while FML buffers do not directly support the `dec_t type`, you can place decimal values in string fields and map these to `dec_t` fields within VIEW definitions.

Translation Rules for Strings

When planning the interaction between the ATMI platform and host applications, consideration must be given to the programming languages in which the applications are written. A character string is represented differently in the COBOL language than in the C language and associated the ATMI platform VIEW buffer. Listing 5-1 demonstrates the three ways that the same two strings are coded (`string1` and `string2`).

Listing 5-1 Three Representations of Strings

```
C Structure:

struct text

{
       char   rbufsize[5];
       char   testnum[2];
       char   sendnum;
       char   sysid[4];
       char   textfld[10];
       char   string1[10];
       char   string2[16];
};
```

VIEW text

```
#type         cname         fbname count  flag   size   null
char          rbufsize      -      5      -      -      -
char          testnum       -      2      -      -      -
char          sendnum       -      1      -      -      -
char          sysid         -      4      -      -      -
char          textfld       -      10     -      -      -
string        string1       -      1      -      10     -
string        string2       -      1      -      16     -
END
```

COBOL Record

```
       01  TEXT.
           05  RBUFSIZEPIC X(5).
           05  TEXTNUMPIC X(2).
           05  SENDNUM PIC X.
           05  SYSID   PIC X(4).
           05  STRING1 PIC X(9).
           05  STRING2PIC X(15).
```

The listing shows that, in the C structure and VIEW buffer, the sizes of `string1` and `string2` are represented as 10 and 16 characters, respectively. However, in the COBOL record, the sizes are 9 and 15 characters, respectively. This incompatibility can cause code misalignment between C and COBOL programs if not anticipated in the source code.

To avoid such incompatibilities, the gateway provides a software option to control the mapping of string data between C and COBOL applications. This option enables you to automatically compensate for the differences in null termination and padding characteristics of the two languages.

Note: The option affects string fields in the ATMI platform VIEW buffers only. STRING buffers are not affected by this switch.

Setting the Option to Perform String Transformation

To set the string transformation option, use the `CLOPT` parameter when you configure the gateway server (GWSNAX) definition in the `UBBCONFIG` file. If you set the `-t` option of the `CLOPT` parameter to one of the values listed in Table 5-4, the gateway performs the corresponding string transformation. Use the following syntax format:

```
CLOPT="-- -t {number}"
```

In this parameter, arguments and options are defined in the following way:

`CLOPT` specifies the ATMI parameter which enables you to provide command-line options in a server definition.

`--` marks the end of system-recognized arguments and the start of arguments passed to a subroutine within the server. This option is required if you supply application-specific arguments, such as the `-t` option, to the server.

`-t` is the eAM option to establish C-to-COBOL string transformation.

`{number}` indicates the type of string transformation the gateway performs as shown in Table 5-4.

Note: If you do not set the `-t` option of the `CLOPT` parameter in your server definition, by default the gateway performs no string transformation.

Table 5-4 C to COBOL String Transformation

CLOPT -t Parameter Value	ATMI Application Language	Host Application Language
Not Set	No string transformation established	
1	C	COBOL
2	COBOL	C
3	C	C
4	COBOL	COBOL

These options function in the following ways:

Option value 1:
: For outbound messages to the host, C string fields are converted to COBOL string fields. All available characters, up to the defined length of the string and beginning with the null character, are converted to spaces and the length of the field is reduced by one.
: For inbound messages from the host, COBOL string fields are converted to C string fields, trailing blanks are converted to null characters (zero value) and the length of the field is increased by one.

Option value 2:
: For outbound messages to the host, COBOL string fields are converted to C string fields, trailing blanks are converted to null characters (zero value) and the length of the field is increased by one.
: For inbound messages from the host, C string fields are converted to COBOL string fields. All available characters, up to the defined length of the string and beginning with the null character, are converted to spaces and the length of the field is reduced by one.

Option values 3 and 4:
: No string transformations are made between programs written in compatible languages.

Example:

The following example of a server definition uses the switch to establish string transformations between an ATMI application written in C and a host application written in COBOL.

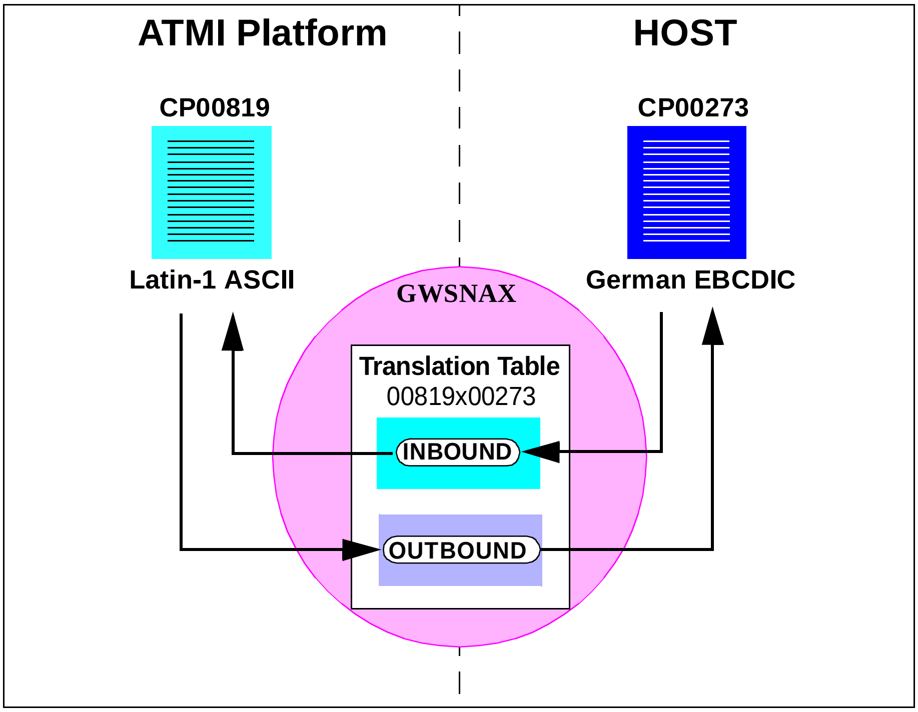
```
*SERVERS

GWSNAX SRVGRP=GROUP1 SRVID=5 CLOPT=”-A -- -t 1”
```

Code Page Translation Tables

The eAM software includes translation tables that enable conversions between ASCII character sets and EBCDIC character sets. The code pages provide 12 standardized tables to facilitate operations between ATMI applications using the Latin-1 ASCII code set (CP-00819) and host applications using a national language code set.

Each translation table consists of two mapping tables, one for outbound conversions (ATMI platform-to-host) and one for inbound conversions (host to ATMI platform). You do not have to specify the direction of a translation. You only need to determine the national language in which the host application is written. Figure 5-1 illustrates code page translation.

Figure 5-1 eAM Code Page Translation



The figure demonstrates how an ATMI application using the Latin-1 ASCII code page CP-00819 character set operates with a host application using German EBCDIC code page CP-00273. The eAM translation table 00819x00273 provides both the inbound and outbound conversions.

Specifying a Translation Table

To designate the translation table for your applications, make an entry in the ATMI platform `DMCONFIG` file definition for each remote domain. Use the `CODEPAGE` parameter with the following format:

For example:

```
*DM_REMOTE_DOMAINS

BEAS TYPE=SNAX CODEPAGE=“cpname”
```

In this parameter, `cpname` identifies the translation table for the remote domain, from Table 5-5. It must be enclosed by double quotes.

Table 5-5 lists the translation tables provided with eAM software.

Table 5-5 eAM Code Page Translation Tables

Country	File Name	ASCII Code Set	EBCDIC Code Set
N/A	none	No translation	No translation
ATMI platform default [a]	tuxedo	ATMI ASCII	ATMI EBCDIC
United States	00819x00037	CP-00819 [b]	CP-00037
Great Britain	00819x00285	CP-00819	CP-00285
France	00819x00297	CP-00819	CP-00297
Portugal	00819x00860	CP-00819	CP-00860
Spain	00819x00284	CP-00819	CP-00284
Belgium	00819x00500	CP-00819	CP-00500
Germany	00819x00273	CP-00819	CP-00273
Finland	00819x00278	CP-00819	CP-00278
Sweden	00819x00278	CP-00819	CP-00278
Latin-1	00819x01047	CP-00819	CP-01047
Latin-2	00912x00870	CP-00912	CP-00870

a. The default ATMI ASCII and EBCDIC code pages are slightly different from CP-00819 and CP-00037.
b. CP-00819 is exactly equivalent to ISO-8859-1 (also called Latin-1 ASCII), and is used as the ASCII code page in all of the countries listed in this table.

How the Translation Tables Work

The eAM translation tables are based on IBM-defined code sets. At start up, the gateway loads a translation table for each remote domain.

You can modify any of the tables to suit your application translation needs, except the default ATMI tables, which are hard-coded. Refer to Appendix D, "Code Page Translation Tables" for detailed table contents. You must restart the gateway to change any translation table definitions.

Note: Replicas of the default ATMI translation tables are included with your product software. These tables are provided for you to modify, if desired. They are not the actual default tables. You cannot modify the default ATMI tables.

The eAM translation tables are located in the following sub-directory:

```
$TUXDIR/udataobj/codepage
```

If no `CODEPAGE` specification is made for a remote domain, the eAM software uses the ATMI default translation tables. If the software cannot find the translation table file, it generates a message `2241:ERROR Unable to access codepage table` with a reason code and the gateway fails to start. Refer to this message in Appendix B, "Error Messages" for explanations of the reason codes.

Listing 5-2 depicts entries defining one local domain (CIXA) and two remote domains (CISA and IMSA). In all cases, the assumption is made that the local domain uses ASCII code page CP-00819. In the example, the two remote domains use the German and French EBCDIC code pages CP-00273 and CP-00297, respectively.

Listing 5-2 Code Page Definition Example

```
# DMCONFIG
*DM_LOCAL_DOMAINS
CIXA TYPE=SNAX
*DM_REMOTE_DOMAINS
CISA TYPE=SNAX CODEPAGE=”00819X00273”
IMSA TYPE=SNAX CODEPAGE=”00819X00297”
```

Using Information Integrator

The gateway supports use of the BEA eLink Information Integrator to perform data mapping. The following sections explain the data mapping tool and how it is configured.

The data mapping tool supported by eAM is eLink Information Integrator. This data mapping tool can be configured to map Field Manipulation Language (FML) buffers to COBOL data buffers. The eLink Information Integrator implements this functionality using ATMI platform and eAM gateway configuration settings.

The following sections discuss:

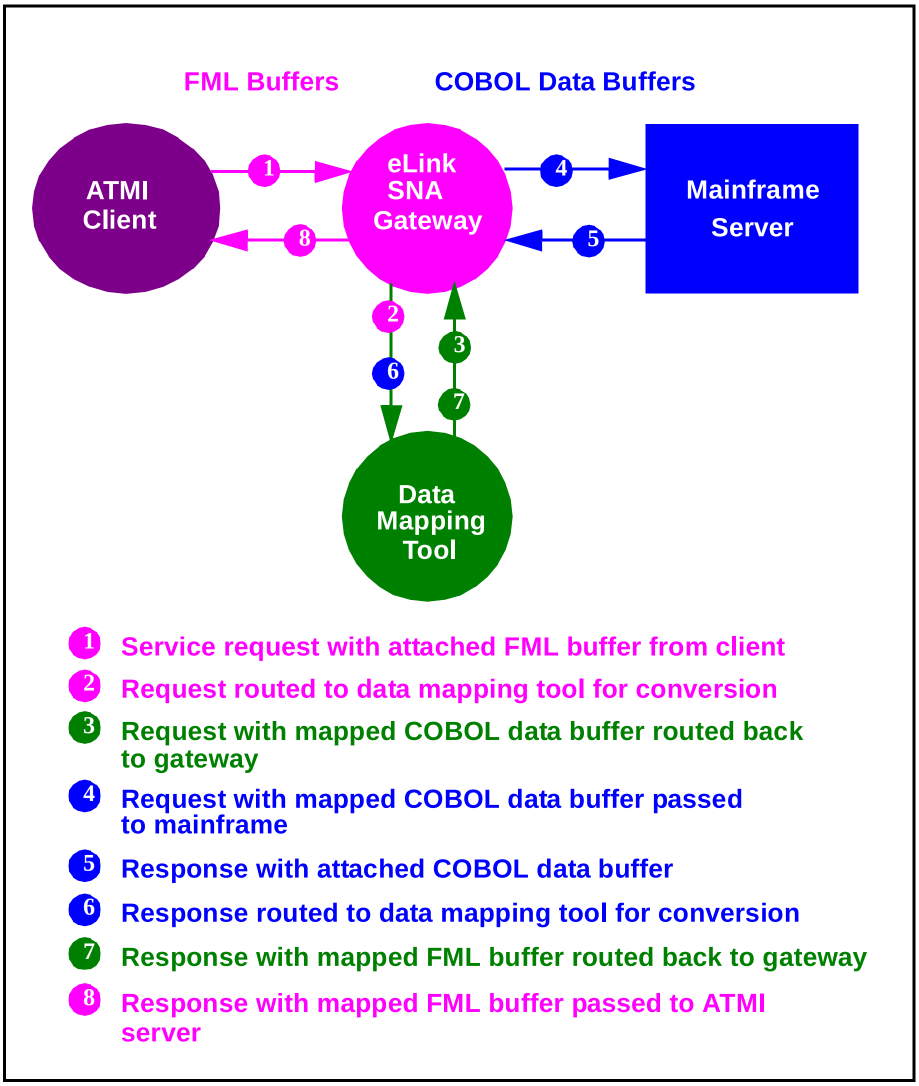
- An overview of the data mapping process flow
- Platform compatibility between the eLink Information Integrator and the gateway
- Configuration settings necessary to implement data mapping

About Data Mapping

When an ATMI client sends a service request with an attached FML buffer to a mainframe server through the eAM gateway, the request can be routed through the data mapping tool. The data mapping tool maps the FML buffer data to a corresponding COBOL data buffer. The gateway then passes the COBOL data buffer to the mainframe. The response from the mainframe is routed in the reverse manner and the data mapping tool maps from COBOL data back to FML data. This method of conversion applies in both directions, no matter which side initiates the transaction. See Figure 5-2 and Figure 5-3 for the Data Mapping Process Flow.

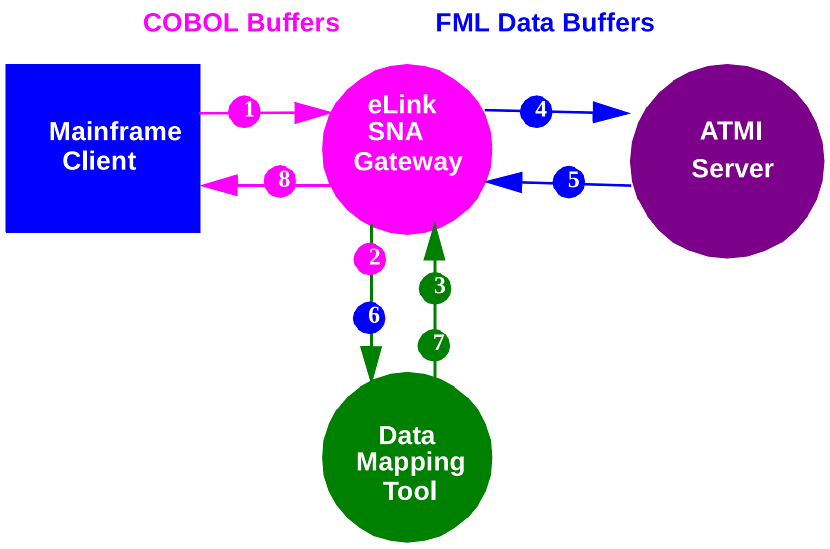
Before using the eLink Information Integrator, you must configure the ATMI server and the eAM gateway.

Figure 5-2 Data Mapping Process Flow: Request Initiated by ATMI Client



1. Service request with attached FML buffer from client
2. Request routed to data mapping tool for conversion
3. Request with mapped COBOL data buffer routed back to gateway
4. Request with mapped COBOL data buffer passed to mainframe
5. Response with attached COBOL data buffer
6. Response routed to data mapping tool for conversion
7. Response with mapped FML buffer routed back to gateway
8. Response with mapped FML buffer passed to ATMI server

Figure 5-3 Data Mapping Process Flow: Request Initiated by Mainframe



1. Service request with attached COBOL buffer from mainframe
2. Request routed to data mapping tool for conversion
3. Request with mapped FML data buffer routed back to gateway
4. Request with mapped FML data buffer passed to client
5. Response with attached FML data buffer
6. Response routed to data mapping tool for conversion
7. Response with mapped COBOL buffer routed back to gateway
8. Response with mapped COBOL buffer passed to mainframe

Setting Up the eLink Information Integrator in the UBBCONFIG File

As shown in Listing 5-3, the eAM gateway must be configured to recognize the eLink Information Integrator to do the data mapping. The `-e` gateway option in the `CLOPT` parameter of the `SERVERS` section in the `UBBCONFIG` file sets up the gateway to recognize this external tool. The value for eLink Information Integrator data mapping is `-eII`. Listing 5-3 is a sample `UBBCONFIG` file that contains this definition.

Defining the IIServer

Define the `IISERVER` as the server responsible for the data mapping operation. To define this server, add the `IISERVER` information in the SERVERS section of the `UBBCONFIG` file.

Note: You must specify `IISERVER` as the file to execute for the data mapping process. Within this server definition, you must also specify the `-s` option in the `CLOPT` parameter.

Advertising the Services for Data Mapping

Advertise the service for data mapping by defining it in the `SERVICES` section of the `UBBCONFIG` file as shown in Listing 5-3.

Note: The name of each of the services must match the map names in the Information Integrator server configuration. For additional information, see “Configuring Information Integrator and the IISERVER” in the *BEA Information Integrator User Guide*.

Sample UBBCONFIG File for Data Mapping

Listing 5-3 is a sample `UBBCONFIG` file for Windows NT in which the eLink Integrator is specified as an alternate data mapping tool. In this sample, the `elinkmerc` server is defined in the SERVERS section with the required `CLOPT -s` option specified. Also, the `-eII` option is specified in the `CLOPT` parameter of the gateway server definition.

Listing 5-3 Sample `UBBCONFIG` File for Data Mapping

```
*RESOURCES

IPCKEY         123791
DOMAINID       simpapp
MASTER         simple

*MACHINES

My Machine
       LMID          = simple
       TUXDIR        = "\tuxedo"
       TUXCONFIG     = "\myappdir\tuxconfig"
       APPDIR        = "\myappdir"
       FIELDTBL      = "sample.fml"
       FIELDTBL32    = "sample.fml"
       FLDTBLDIR     = "\myappdir"
       FLDTBLDIR32   = "\myappdir"
       ULOGPFX       = "\myappdir\ULOG"
                                   # LD_LIBRARY_PATH=\dio
                                   # SHLIB_PATH=\dio
                                   PATH=\dio


*GROUPS

eLINK
       LMID=simple   GRPNO=1

*SERVERS

DEFAULT:
       CLOPT="-A"

IISERVER
       SRVGRP=eLINK         SRVID=10
       REPLYQ=N
       CLOPT="-sFML2COB:XLATE_SERVICE -sCOB2FML:XLATE_SERVICE --
-WUD -TIO -AE"

GWSNAX
        SRVGRP=eLINK   SRVID=21
        CLOPT="-A -- -eII"


*SERVICES
II_SERVICE
```

```
FML2COB
COB2FML

*ROUTING
```

Setting Up eLink Information Integrator in the DMCONFIG File

Before running the gateway with eLink Information Integrator, you must edit the ATMI platform `DMCONFIG` file to specify the local and remote services and buffer types used for data mapping, as shown in Listing 5-4.

In the sample configuration, local service `INFML` and remote service `OUTFML` contain FML views that eLink Information Integrator translates to mainframe COBOL data buffers (`FML2COB`, specified in the `INBUFTYPE` definitions). These services also contain FML views that eLink Information Integrator translates to FML buffers (`COB2FML`, specified in the `OUTBUFTYPE` definitions).

Listing 5-4 Sample `DMCONFIG` File for Data Mapping

```
*DM_LOCAL_DOMAINS

LOCAL           GWGRP=GROUP
                TYPE=IDOMAIN
                DOMAINID="LOCAL"

*DM_REMOTE_DOMAINS

REMOTE          TYPE=IDOMAIN
                DOMAINID="REMOTE"

*DM_LOCAL_SERVICES

INFML           RNAME="TV16X"
                INBUFTYPE="FML:FML2COB"
                OUTBUFTYPE="FML:COB2FML"

*DM_REMOTE_SERVICES
```

```
OUTFML          RDOM=REMOTE
                LDOM=LOCAL
                INBUFTYPE="FML:FML2COB"
                OUTBUFTYPE="FML:COB2FML"
                RNAME="ZOUTFML"
```

Specifying Inbound Encoding and Decoding Services

For transactions originated by the mainframe, configure the `DM_LOCAL_SERVICES` section of the `DMCONFIG` file. See Listing 5-4 for an example.

The advertised input service (`INFML`) includes an `INBUFTYPE` and an `OUTBUFTYPE.` Use the `INBUFTYPE` to specify the service for encoding FML buffers to COBOL data buffers (`FML2COB`). Use the `OUTBUFTYPE` to specify the service for decoding COBOL data buffers to FML buffers (`COB2FML`).

In terms of request/response exchanges, the settings result in the following conditions:

1. The local service handles requests from the mainframe using an `OUTBUFTYPE` that the eLink Information Integrator decodes from COBOL to FML.
2. The local service handles responses to the mainframe with an `INBUFTYPE` that the eLink Information Integrator encodes from FML to COBOL.

Note: If the `IISERVER` option is not set, the `INBUFTYPE` and `OUTBUFTYPE` parameters operate as described in the `dmconfig` section of Appendix A, "Reference Pages."

Specifying Outbound Encoding and Decoding Buffers

For transactions originated by the ATMI platform, configure the `DM_REMOTE_SERVICES` section of the `DMCONFIG` file. See Listing 5-4 for an example.

The advertised output service (`OUTFML`) includes an `INBUFTYPE` and an `OUTBUFTYPE.` Use the `INBUFTYPE` to specify the service for encoding FML buffers to COBOL data buffers (`FML2COB`). Use the `OUTBUFTYPE` to specify the service for decoding COBOL data buffers to FML buffers (`COB2FML`).

In terms of request/response exchanges, these settings result in the following conditions:

1. The remote service handles requests from the ATMI platform using an `INBUFTYPE` that eLink Information Integrator encodes from FML to COBOL.
2. The remote service handles responses to the ATMI platform with an `OUTBUFTYPE` that eLink Information Integrator decodes from COBOL to FML.

Note: If the `IISERVER` option is not set, the `INBUFTYPE` and `OUTBUFTYPE` parameters operate as described in the `dmconfig` section Appendix A, "Reference Pages."

6 APPC/IMS Programming Considerations

This section is intended for application programmers who implement and integrate ATMI platform and host enterprise applications using APPC/IMS programs. The application programmer in the Information Management System (IMS) environment can use implicit IMS programming techniques.

Note: All references to ATMI files, functions, and documentation apply to Tuxedo, eLink Platform, and WebLogic Enterprise files, functions, and documentation.

This section discusses the following topics:

APPC/IMS Overview

APPC/IMS allows application programs using APPC protocols to enter IMS transactions from LU 6.2 devices supporting APPC. APPC/IMS also provides an environment that enables remote LU 6.2 devices to enter IMS local and remote transactions. In this environment, IMS application programs can insert transaction output to LU 6.2 devices without requiring coding changes to existing application programs and new application programs can make full use of existing LU 6.2 facilities. Applications enter transactions using an implicit or explicit API.

Implicit API

The implicit API can be a useful simplification for many applications. While it does not provide all the existing LU6.2 capabilities, this API provides additional functions, such as message queueing and automatic asynchronous message delivery.

Using the IMS application programming base with the implicit API, you can write transactional applications that do not have Common Programming Interface for Communications (CPI-C) calls. IMS generates all the CPI-C calls for you. The application interaction is strictly with the IMS message queue.

The implicit API accesses an APPC conversation indirectly. It uses the standard DL/I calls (GU, ISRT, PURG) to send and receive data. The implicit API allows non-LU 6.2 specific applications to use LU 6.2 transactional protocols, using new and changed DL/I calls (CHNG, INQY, SETO).

The implicit API creates asynchronous LU 6.2 output by using alternate PCBs referencing LU 6.2 destinations. The DL/I CHNG call can supply parameters to specify an LU 6.2 destination. Default values substitute for omitted parameters.

An application program can use the implicit API to retrieve the current conversation attributes, such as the conversation type (basic or mapped), the sync_level, and whether it is asynchronous or synchronous.

Explicit API

An IMS application program can use the explicit API to issue the CPI-C calls directly. The explicit API is useful with remote LU 6.2 systems that have incomplete LU 6.2 implementations, or that are incompatible with the IMS implicit API support.

The explicit API can be used by any IMS application program to access an APPC conversation directly. IMS resources are available to the CPI-C driven application program only if the application issues the APSB (Allocate_ PSB) call. The CPI-C driven application program must use the CPI-RR `SRRCMIT` and `SRRBACK` verbs to initiate an IMS sync point or backout.

APPC/IMS Programming

The eAM system supports non-transactional and transactional IMS servers using either the implicit APPC support for IMS or the explicit APPC interface using APPC/MVS calls from a user application. Any IMS program that gets messages from, and puts messages into, the IMS message queue can be used without change as either a client or server.

To use the implicit APPC capabilities of IMS, you must modify the `APPCMxx` file in the `SYS1.PARMLIB` library provided with your eAM software. The configuration parameters in this file associate the LU with the IMS scheduler. You must identify the LU representing the application name used by eAM to access the IMS region and the IMS system ID which provides scheduling for inbound requests. Be sure to discuss with mainframe support personnel the changes you make to the APPCMxx file.

Non-Transactional Application Programming

Listing 6-1 is an example of a non-transactional program. In this example, the VTAM application major node is designated to be MVSLUO1 and the scheduling facility is designated to be the IMS control region IVP4.

Listing 6-1 APPCM File in SYS1.PARMLIB Library (Example Only)

```
SYS1.PARMLIB(APPCMxx)

LUADD ACBNAME(MVSLU01) BASE TPDATA(SYS1.APPCTP),
SCHED(IVP4),
SIDEINFO DATASET(SYS1.APPCSI)

SYS1.VTAMLST(MVSLU01)

MVSLU01 APPL  ACBNAME=MVSLU01,       ACBNAME FOR APPC             C
              APPC=YES,                                           C
              AUTOSES=0,                                          C
              DDRAINL=NALOW,                                      C
              DLOGMOD=APPCHOST,                                   C
              DMINWNL=3,                                          C
              DMINWNR=3,                                          C
              DRESPL=NALLOW,                                      C
              DSESLIM=6,                                          C
              LMDENT=19,                                          C
              MODETAB=APPCTAB,                                    C
              PARSESS=YES,                                        C
              SECACPT=CONV,                                       C
              SRBEXIT=YES,                                        C
              VPACING=1
```

The job that starts the IMS subsystem should have the APPC parameter set to `Y`. The example in Listing 6-2 illustrates such a job, but is not intended to be used under actual conditions. Use your own custom job for starting IMS.

Listing 6-2 IMS Subsystem Start Job (Example Only)

```
PROC RGN=2000K,SOUT=A,DPTY='(14,15)',
          SYS=,SYS1=,SYS2=,
          RGSUF=IV1,PARM1=APPC=Y,PARM2=,APPLID1=IMS61CR1,AOIS=R
IEFPROC EXEC PGM=DFSMVRC0,DPRTY=&DPTY,
          REGION=&RGN,
          PARM='CTL,&RGSUF,&PARM1,&PARM2,&APPLID1,&AOIS'
*
*
* THE MEANING AND MAXIMUM SIZE OF EACH PARAMETER
* IS AS FOLLOWS:
*
```

```
********** CONTROL REGION SPECIFICATIONS ***********
****************************************************
*   RGSUF   XXX   EXEC PARM DEFAULT BLOCK SUFFIX FOR
*                 MEMBER DFSPBXXX.
****************************************************
*
*  PARM1 , PARM2 PARAMETERS BOTH ARE USED TO SPECIFY
*  CHARACTER STRINGS THAT CONTAIN IMS KEYWORD
*
*  PARAMETERS. I.E. PARM1='AUTO=Y,PST=222,RES=Y'
*
*
*   APPC    X     Y = ACTIVATE APPC/IMS
*                 N = DO NOT ACTIVATE APPC/IMS
```

Transactional Application Programming

Listing 6-3 is an example of a transactional VTAM program. The inclusion of the LU definition `SYNCLVL=SYNCPT` (shown in bold) makes the program transactional.

Note: You should include the `ATNLOSS=ALL` parameter value whenever you use the `SYNCLVL=SYNCPT` definition.

Listing 6-3 Sample VTAM LU Definition

```
MVSLU01 APPL   ACBNAME=MVSLU01,        ACBNAME FOR APPC       C
               APPC=YES,                                      C
               AUTOSES=0,                                     C
               DDRAINL=NALOW,                                 C
               DLMOD=APPCHOST,                                C
               DMWNL=5,                                       C
               DMINWNR=5,                                     C
               DRESPL=NALLOW,                                 C
               DSESLIM=10,                                    C
               LMDENT=19,                                     C
               MODETAB=APPCTAB,                               C
               PARSESS=YES,                                   C
               SECACPT=CONV,                                  C
               SRBEXIT=YES,                                   C
               SYNCLVL=SYNCPT,                                C
```

```
ATNLOSS=ALL,                                                    C
VPACING=1
```

Sample Transaction Programs

The following eAM transactional test programs are installed in the ATMI platform installation in the subdirectory `eLink/sna/simpapp`:

- `simpims.c` is a simple ATMI client used to invoke both the sample IMS server programs. It takes a data string and service name as inputs. It invokes the service and passes the input data string.
- `IMPIMSSV.cbl` is a simple IMS echo server. It reads data from the IMS message queue and writes the same data in response. It is intended to be used as an implicit SNA example.
- `EXPIMSSV.c is` an IMS server transaction using explicit CPI-C calls. It is written for sync level 2 use. The program uses an IBM sample database, `IVPDB2`. The program displays, adds, and deletes records from the database, based on an input string. Sample input strings are documented in the source.
- `BEAWTOR.asm` is an assembler subroutine used by `EXPIMSSV.c` to write messages to the MVS console log.

7 Integrating eAM with Crossplex

BEA eLink for Mainframe products use Intersystem Communications Distributed Program Link to access CICS applications passing a `COMMAREA` in the buffer between the ATMI platform and the CICS application. Applications need to be architected to separate the business logic from the presentation logic. The ATMI application will "fill in" the required information that was once done by the CICS 3270 terminal input and link to the program defined in the eAM configuration file. The eAM software also has the ability to support conversations (DTP). All products are bidirectional, also supporting requests originating from CICS or IMS.

Integrating eAM with CrossPlex by SofTouch allows you to address the following concerns:

- Major overhaul to build distributed applications
- Unavailable or inaccessible mainframe source code
- Rapid integration

The eAM gateway can be easily configured to link (DPL) to CrossPlex instead of the original intended application, passing CrossPlex all control information necessary to dynamically initiate the CICS transaction through a FEPI virtual terminal defined by CrosssPlex. This scenario requires no host code modification.

CrossPlex Architecture

CrossPlex offers a complete development and runtime tool kit for re-engineering host-based applications, quickly transforming them into applications for today's technology. CrossPlex leverages applications that already work, eliminating the risk and expense of re-programming. CrossPlex offers the following features:

- Preserves and reuses the assets of your business logic and the investments made in existing applications
- Offers no down time, no re-engineering, no migration, and no costly conversion.
- Provides operator with a highly customized and efficient interface
- Requires no change to existing application code
- Integrates multiple applications into one presentation
- Maintains current speed, security, scalability, and reliability
- Future-proofs by allowing integration to any technology — present and future

With CrossPlex, traditional systems can be converted to web applications, Java applications, or simply integrate existing online systems with distributed environments. These applications, referred to as *legacy*, work extremely well. Their only drawback is the customer interface. Users may need to pass through as many as 17 different screens to activate a transaction. CrossPlex allows you to take these screens, convert them into one HTML or Java screen, and point and click.

CrossPlex provides definition tools to capture screen datastreams and create CrossPlex Object Definitions. These object definitions are used in script creation and execution. They define the 3270 datastream.

A Command Stream or script is a navigation path and data extraction directive for existing applications. This method is used when eAM makes a request that requires information contained on several application screens. A single request sent to CrossPlex invokes several 3270 transactions through the use of a script and collects the information to fulfill the front-end request (referred to as "many-to-one").

For example, an existing 3270 type on-line application may start with a menu screen, followed by a screen which accepts the input of a client key, which is followed by a series of three screens that are invoked by PF keys, each displaying specific information about the client. Using the command stream method, CrossPlex processes all the interactions of the on-line application and gathers all information in the process. Only one complete data packet is sent to eAM.

The CrossPlex Visual Application Developer (VAD) is a Java application that allows you to create and maintain CrossPlex command streams. Visual Application Developer starts with a recorded queue of legacy 3270 screen objects and provides the capability to create a CrossPlex Script from these screen objects using a user-friendly GUI interface.

The Visual Application Developer allows the developer to visually see the flow of a series of 3270 transactions that will be driven by a script. The developer views the 3270 transaction flow, selects fields to be captured or used for subsequent input, adds any required conditional logic, or comments, and then requests VAD to generate a completed script, which is saved on the mainframe.

The end result is an executable CrossPlex command stream, ready for testing. This generated script can be reloaded with VAD for changes and alterations, or it can be edited with the 3270-based script editor.

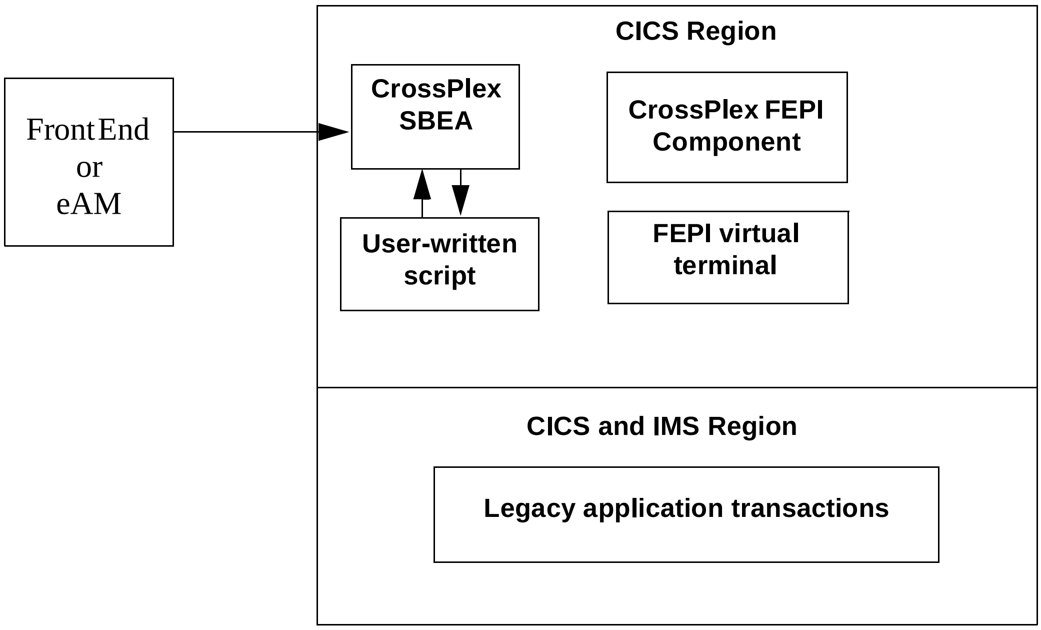
CrossPlex also offers advantages that involve 3270 transaction execution. With CrossPlex, all 3270 transactions execute data transmissions "in memory." No network communication is involved while the 3270 application is operating, no matter how many transactions are executed. The only transmission on the network is the final result sent as a single response. The result is that network traffic and bandwidth usage is reduced with CrossPlex. The 3270 transactions respond to CrossPlex faster than they do when a user is executing them at a terminal.

This ability of CrossPlex to suppress network traffic is one reason the two-tiered approach is preferable to a three-tiered alternative. With a three-tiered solution, a server computer must communicate with the mainframe, individually executing 3270 transactions one at a time, transmitting each across the network to the server computer. With CrossPlex, all sending and receiving of 3270 datastreams is maintained on the mainframe, before anything is placed on the network or sent back to eAM.

The combination of eAM and CrossPlex enables customers to efficiently create, test, and deploy transactional e-business applications that can be scaled as business needs demand.

CrossPlex installs and operates on the OS/390 mainframe. It runs in a CICS region but has the ability to make VTAM connections to other systems, such as IMS. It makes this connections by creating a virtual terminal, using an IBM component of CICS called Front End Programming Interface (FEPI). Figure 8-1 illustrates the process.

Figure 7-1 CrossPlex Architecture



This section contains a scenario that shows how to develop an ATMI client that invokes a CrossPlex script on the mainframe. Similar techniques may be used to interface to other third-party products.

Note: Although the sample code in this section represents typical applications, it is intended for example only and is not supported for actual use.

Note: All discussions of ATMI files, functions, and documentation apply to Tuxedo, eLink Platform, and WebLogic Enterprise files, functions, and documentation.

The following tasks are required to invoke a CrossPlex script on the mainframe:

- Task 1: Create a CrossPlex script.
- Task 2: Create a view definition that describes the application data.

- Task 3: Code your client program.
- Task 4: Configure eAM to talk to the CrossPlex installation on your mainframe.
- Task 5: Use the Application.

Task 1: Create a CrossPlex script.

A CrossPlex script provides the business logic to execute one or more 3270 transactions running on the mainframe. Transactions in any VTAM system, such as CICS or IMS, can be accessed. When a script executes in CrossPlex, it usually requires some input data, such as customer number, or part number. This inbound data is passed from the application in a container called a record definition.

During execution, a script will select and optionally reformat data from the screen displays of the executed 3270 transactions. This selected data will be returned to the application in an outbound record definition.

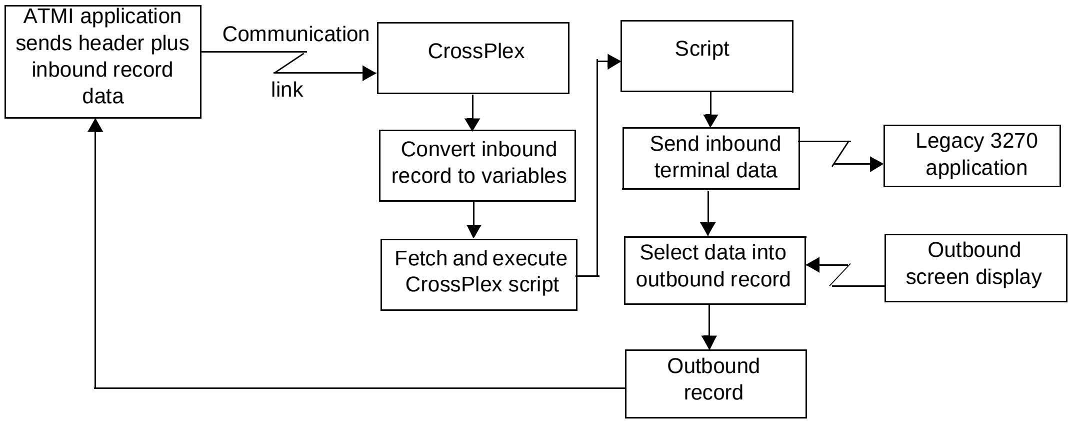
Note: Record definitions do not necessarily conform to any known data record in a file. A record definition is simply a description of a series of data fields being passed to and from a script.

Record definitions are created with the CrossPlex development system. An online editor is used to define each field in the record, along with its length and type (alpha, numeric, binary, packed). A single record definition may be used for both inbound and outbound data, or two definitions may be used.

Another of the CrossPlex development tools will create a COBOL copybook, using a record definition as input. The generated copybook is stored in a PDS member, where it can be copied into your application program as needed.

Figure 7-2 illustrates the processing flow from the eAM front end to retrieve data from one or more mainframe transactions.

Figure 7-2 Processing Flow from eAM to Mainframe Transactions

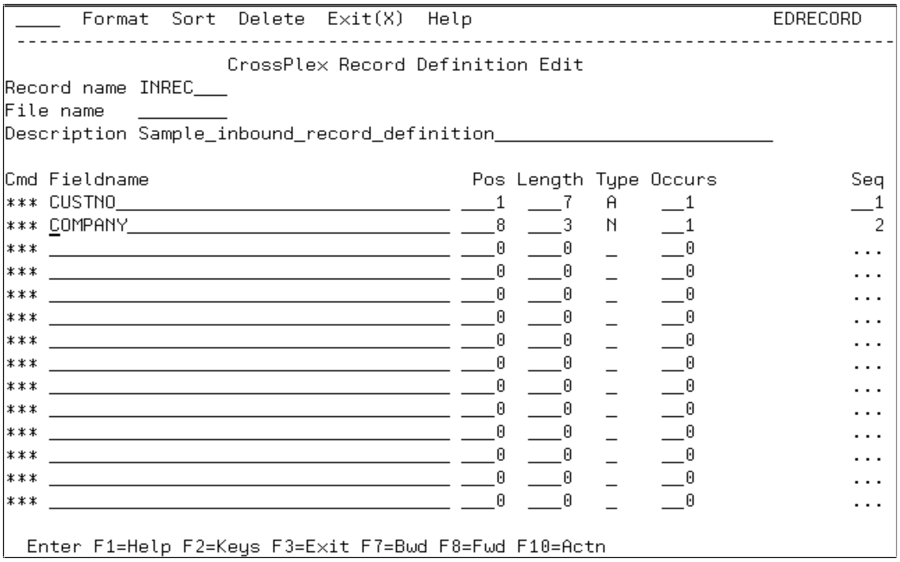

ATMI application sends header plus inbound record data
Communication link
CrossPlex
Convert inbound record to variables
Fetch and execute CrossPlex script
Script
Send inbound terminal data
Legacy 3270 application
Select data into outbound record
Outbound screen display
Outbound record

Step 1: Prepare Inbound Record Definition

Assign a record name and description, then define each data field to be passed to the CrossPlex script. The process of defining a record definition is described in detail in the *CrossPlex Middleware Programmer's Guide*.

To illustrate, assume the mainframe application is a simple name/address display, which requires a customer number and company number as input. For this example, the inbound and outbound record definition will be different, though the same record definition can be used for both. Figure 7-3 shows how the inbound record would appear.

Figure 7-3 Inbound record illustration


```
 ____  Format  Sort  Delete  Exit(X)  Help                           EDRECORD
 ------------------------------------------------------------------------------
                    CrossPlex Record Definition Edit
Record name INREC___
File name   ________
Description Sample_inbound_record_definition_________________________

Cmd Fieldname                                  Pos Length Type Occurs          Seq
*** CUSTNO______________________________     ___1  ___7    A    __1           __1
*** COMPANY_____________________________     ___8  ___3    N    __1             2
*** ____________________________________     ___0  ___0    _    __0           ...
*** ____________________________________     ___0  ___0    _    __0           ...
*** ____________________________________     ___0  ___0    _    __0           ...
*** ____________________________________     ___0  ___0    _    __0           ...
*** ____________________________________     ___0  ___0    _    __0           ...
*** ____________________________________     ___0  ___0    _    __0           ...
*** ____________________________________     ___0  ___0    _    __0           ...
*** ____________________________________     ___0  ___0    _    __0           ...
*** ____________________________________     ___0  ___0    _    __0           ...
*** ____________________________________     ___0  ___0    _    __0           ...
*** ____________________________________     ___0  ___0    _    __0           ...
*** ____________________________________     ___0  ___0    _    __0           ...

  Enter F1=Help F2=Keys F3=Exit F7=Bwd F8=Fwd F10=Actn
```

The inbound data required by the mainframe transaction is CUSTNO, a seven-byte alphanumeric field beginning in position one of the record, and COMPANY, a three-byte numeric field beginning in position eight.

Step 2: Create a Copybook of the Inbound Record Definition

Store the generated copybook in a PDS member where you can easily copy it to your development system. For a complete description of the process of creating a COBOL Copybook from a record definition, refer to the *CrossPlex Middleware Programmer's Guide*.

Continuing with the same example, a COBOL copybook generated from the previously illustrated record definition, INREC, would appear as follows:

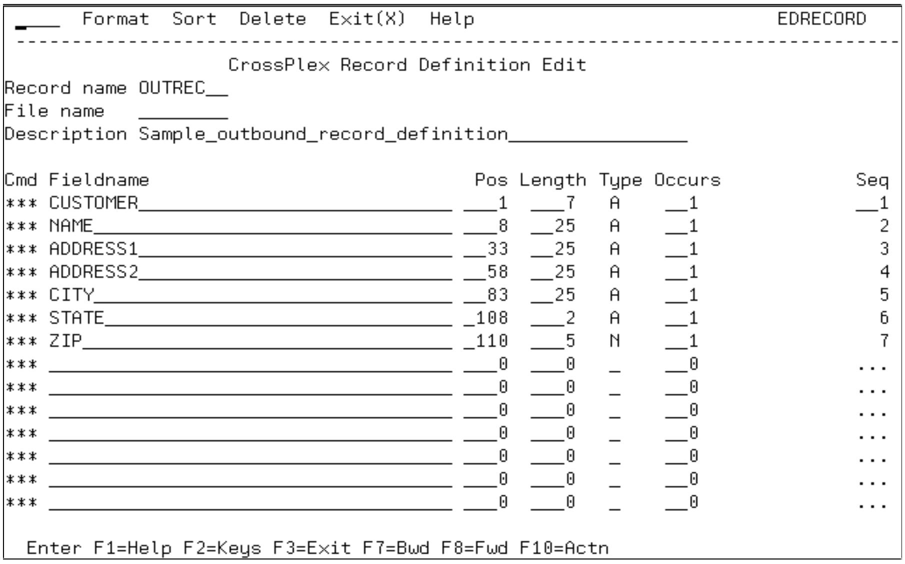
```
*****************************************************************
*      INREC  - Sample inbound record definition*
*****************************************************************
  01   INREC-START.
       05  INREC-CUSTNOPIC X(007).
       05  INREC-COMPANYPIC 9(003).
```

Step 3: Create an Outbound Record Definition and Copybook

If the outbound data is to use a different record format from the inbound, repeat steps 1 and 2 to prepare the outbound record definition and copybook.

For this example, the outbound record definition and copybook would appear as in Figure 7-4.

Figure 7-4 Outbound Record Definition



```
 ____  Format  Sort  Delete  Exit(X)  Help                        EDRECORD
 ---------------------------------------------------------------------------
                    CrossPlex Record Definition Edit
Record name OUTREC__
File name   ________
Description Sample_outbound_record_definition________________

Cmd Fieldname                             Pos Length Type Occurs           Seq
*** CUSTOMER____________________________ ___1 ___7   A    __1             __1
*** NAME________________________________ ___8 __25   A    __1               2
*** ADDRESS1____________________________ __33 __25   A    __1               3
*** ADDRESS2____________________________ __58 __25   A    __1               4
*** CITY________________________________ __83 __25   A    __1               5
*** STATE_______________________________ _108 ___2   A    __1               6
*** ZIP_________________________________ _110 ___5   N    __1               7
*** ____________________________________ ___0 ___0   _    __0             ...
*** ____________________________________ ___0 ___0   _    __0             ...
*** ____________________________________ ___0 ___0   _    __0             ...
*** ____________________________________ ___0 ___0   _    __0             ...
*** ____________________________________ ___0 ___0   _    __0             ...
*** ____________________________________ ___0 ___0   _    __0             ...
*** ____________________________________ ___0 ___0   _    __0             ...

  Enter F1=Help F2=Keys F3=Exit F7=Bwd F8=Fwd F10=Actn
```

```
*****************************************************************
*       OUTREC - Sample outbound record definition*
*****************************************************************
  01    OUTREC-START.
        05  OUTREC-CUSTOMER   PIC X(007).
        05  OUTREC-NAME       PIC X(025).
        05  OUTREC-ADDRESS1   PIC X(025).
        05  OUTREC-ADDRESS2   PIC X(025).
        05  OUTREC-CITY       PIC X(025).
        05  OUTREC-STATE      PIC X(002).
        05  OUTREC-ZIP        PIC 9(005).
```

Step 4: Prepare the CrossPlex Script

Scripts can be coded using the CrossPlex script editor, or they may be coded on any external editor and imported into the CrossPlex control file. The CrossPlex script language and the process of creating a script are described in the *CrossPlex Middleware Programmer's Guide*.

Note: In the CrossPlex documentation, scripts are also known as command streams and stream objects.

Prepare a script that will navigate through a series of 3270 transactions in the same manner as a terminal operator. The script acts as a virtual operator, performing a log-on to the OLTP system, sending inbound terminal data as if keyed on a keyboard, examining the returned screen display for correct execution, and selecting data from the screen if needed. Any number of transactions may be executed. The script language also provides a method of linking to a user program on the mainframe in order to perform direct retrieval of data that may not be available in a 3270 transaction display.

Continuing with the example of name/address data retrieval, the script might appear as Listing 7-1.

Listing 7-1 CrossPlex Script

```
CALLCPX MSGAREA(NMAD)                      Initiate transaction NMAD.
CALLCPX ROWCOL(05023) DATA(&CUSTNO)        Send CUSTNO to row 5 col 23.
IF ROWCOL(24021) EQ DATA(NOT ON FILE)-     Verify customer record found
      GOTO(NOTFOUND)
SELECT RECORD(OUTREC) -                    Select data from outbound
      ROWCOL(05023) RFIELD(CUSTNO) -             screen into remaining
      ROWCOL(06023) RFIELD(NAME) -               record fields.
      ROWCOL(07023) RFIELD(ADDR1) -
      ROWCOL(08023) RFIELD(ADDR2) -
      ROWCOL(09023) RFIELD(CITY) -
      ROWCOL(10023) RFIELD(STATE) -
      ROWCOL(11023) RFIELD(ZIP)
GOTO(ENDJOB)                               Skip following error routine
NOTFOUND                                   Enter if customer not found
      SELECT RECORD(OUTREC) -              Move zeros to customer number
          DATA(0000000) RFIELD(CUSTNO)
ENDJOB                                     Enter or fall through
      CALLCPX AID(PF3)                     Terminate NMAD transaction
```

Note: This example illustrates row/column addressing of screen data. CrossPlex also provides a method of assigning screen field names to avoid specific row/column references

Step 5: Test and Debug the Script

You can fully test and debug the script that will execute on the mainframe without connecting it to your front-end application. CrossPlex provides a variety of execution and debugging tools to ensure the back-end portion of your application is operating properly.

Once you are satisfied that the script is doing what you want and the returned data is correct, proceed to prepare the front-end of your application and connect the two together.

The process of testing and debugging a script is described in the *CrossPlex Middleware Programmer's Guide*.

Handling the Mainframe Sign-on

Most VTAM systems require the user to sign-on in the target region when first connecting. This procedure is also true when connecting to a target region with CrossPlex. This sign-on requirement can be handled in any one of the following ways:

- Interact with a user sign-on transaction in the script.

  The most common situation, especially for CICS, requires that your script handle the sign-on. Many users have CICS configured so that upon the first connection, the terminal is presented with a sign-on panel that may have been customized for the installation. If this is the case, the first CALLCPX command of the script returns the sign-on screen to the script and a subsequent CALLCPX must send a valid user ID and password. The mainframe sign-in is discussed in the *CrossPlex Middleware Programmer's Guide*.

- Let CrossPlex perform a short-form sign-on.

  Supplying a valid user ID and password in the CrossPlex header will cause CrossPlex to perform a short-form sign-on before sending the first transaction data from the script.

  Note: This method is valid for CICS systems only, and is installation dependent.

The short-form CICS sign-on may be disabled, depending on the user's CICS configuration. This is discussed in the *CrossPlex Middleware Programmer's Guide*.

- Perform a mass log-on at CICS startup.

  With this technique, several FEPI virtual terminals are logged-on when CICS is first started and they remain active until CICS is recycled. If this is done, scripts do not need to be concerned with doing a sign-on at all. See the *CrossPlex Web Enabling Guide* for further discussion.

Task 2: Create a view definition that describes the application data.

This view definition will need to include both the CrossPlex header and your application data. It may be necessary to create two such views, one for the request data and the other for the response. In this example, the user data is all of type character and thus a combined view is usable. The following view file describes the CrossPlex header.

Listing 7-2 CrossPlex Standard Commarea (Tuxedo VIEW)

```
#
#    compiled using "viewc -n cpxview.v"
VIEW CPXVIEW
# type   cname      fbname  count   flag   size   null  comment
char   xpcmd      -       4       -      -      -
long   xpexcep    -       1       -      -      -
short  xpexrc     -       1       -      -      -
short  xpextl     -       1       -      -      -
short  xpexmsg    -       1       -      -      -
short  xpexmof    -       1       -      -      -
char   xpexxp1    -       4       -      -      -
long   xpexxp2    -       1       -      -      -
long   xpexxp3    -       1       -      -      -
long   xpoptpl    -       1       -      -      -
char   xptarg     -       8       -      -      -
char   xpfpool    -       8       -      -      -
char   xpaid      -       6       -      -      -
char   xpinsc     -       8       -      -      -
char   xpotsc     -       8       -      -      -
short  xpcrow     -       1       -      -      -
short  xpccol     -       1       -      -      -
char   xpuser     -       8       -      -      -
char   xppass     -       8       -      -      -
char   xpfnode    -       8       -      -      -
short  xpfconv    -       4       -      -      -
char   xpdbugq    -       8       -      -      -
char   xpassoc    -       8       -      -      -
char   xpmode     -       4       -      -      -
char   xpxlate    -       8       -      -      -
short  xpilng     -       1       -      -      -
short  xpalng     -       1       -      -      -
short  xpolng     -       1       -      -      -
char   xptermop   -       1       -      -      -
char   xpusd      -       1       -      -      -
short  xpusdwt    -       1       -      -      -
char   xpprofa    -       8       -      -      -
char   xpprofk    -       16      -      -      -
char   xpprog     -       8       -      -      -
short  xprows     -       1       -      -      -
short  xpcols     -       1       -      -      -
char   xpstrm     -       8       -      -      -
short  xptimot    -       1       -      -      -
char   xpxltyp    -       1       -      -      -
char   xpdbugs    -       1       -      -      -
long   xpmaptr    -       1       -      -      -
long   xpmalng    -       1       -      -      -
long   xpmdlng    -       1       -      -      -
char   xpmause    -       1       -      -      -
char   xprecnm    -       8       -      -      -
char   xpecho     -       1       -      -      -
char   xpctyp     -       1       -      -      -
char   reserved99 -       41      -      -      -
char   xpmarea    -       1000    -      -      -
END
```

In this view, the field named `xpmarea` is the generic character block that will contain the application data, and may be replaced with specific field definitions for your application.

Process this view with the view compiler using a command such as:

```
viewc -n cpxview.v
```

This command will produce a C header file, `cpxview.h`, that will be used to construct the client program.

Task 3: Code your client program.

The following sample program calls CrossPlex in CICS through eAM SNA. CrossPlex then executes several application transactions while gathering data from them. The data is then returned for this program to display.

Listing 7-3 CrossPlex MiddleWare Demo for eLink TCP using a Tuxedo View.

```
#include <stdio.h>
#include <string.h>
#include "atmi.h"    /* TUXEDO Header File */
#pragma pack(1)
#include "cpxview.h" /* Crosslex Header File (as generated by VIEWC)*/
#include "xplxc.h"   /* CrossPlex Header File (as distributed)      */
#include "mwdrec.h"  /* CrossPlex MiddleWare Demo Record Definition */

/* ===================================================================
* cpxError() - determine if CrossPlex returned an error condtion,
*    if so, return xpexcep after printing error message(s) to stderr
*    else,  return 0
*================================================================= */

int cpxError(struct CPXVIEW * xptr) {
       int i, j;
       int rc;

       rc = (int) xptr->xpexcep;  /* Get the exception code */
       if (rc == XPENTIO) {       /* if last screen was clear screen... */
         rc = 0;                 /* ...we don’t care (for this usage)  */
       }
       if (rc) {
         fprintf(stderr, "CrossPlex Exception code %i\n", rc);
         if (rc <= CPXERRTBLCOUNT) {
           fprintf(stderr, "   %s\n", cpxerrtbl[rc]);
       }
       /* Print error messages returned from CrossPlex (if any) */
       j = xptr->xpolng;
       for (i = 0; j > 0 && i < 5; i++, j -= 72) {
         fprintf(stderr, "%.72s\n", &xptr->xpmarea[i*72]);
       }

     /* Display more information for XSCP errors */
     if (rc == XPECONX) {
       fprintf(stderr, "\nSupplemental messages:\n");
       fprintf(stderr, "   XSCP Function %.4s\n", xptr->xpexxp1);
       j = (int) xptr->xpexxp2;
       fprintf(stderr, "   XSCP EIBRESP   code %i\n", j);
       i = (int) xptr->xpexxp3;
       fprintf(stderr, "   XSCP EIBRESP2  code %i\n", i);

       /* Show descriptive message of common FEPI errors    */
       /* See the IBM CICS Library for FEPI EIBRESP2 values */
       if (j == 16) {          /* EIBRESP(16) = FEPI ERROR  */
         if (i == 30)
           fprintf(stderr, "   FEPI POOL %.8s is not known\n",
               xptr->xpfpool);
         else if (i == 32)
           fprintf(stderr, "   FEPI TARGET %.8s is not known\n",
               xptr->xptarg);
         else if (i == 213)
           fprintf(stderr, "   FEPI request timed out\n");
         else if (i == 215)
           fprintf(stderr, "   FEPI session lost\n");
       } /* end if FEPI Exception */
    } /* End if XSCP Exception */
  }/* end if rc */
  return rc;
} /* end cpxError() */

/* ********************************************************************
*  main()
******************************************************************* */

#if defined(__STDC__) || defined(__cplusplus)
main(int argc, char *argv[])
#else
main(argc, argv)
int argc;
char *argv[];
#endif
{

   struct CPXVIEW * sbuf;              // Pointer to XPCOMM send buffer
   struct CPXVIEW * rbuf;              // Pointer to XPCOMM receive buffer
   XPCOMM xpcomm_init = {XPCOMM_INIT}; // To init. the storage area
   MWDREC * mwdrec;                    // To address returned record
   long  rcvlen;                       // Length of data returned from tpcall()
   int   ret;
   int   i;
   char  c;
   char* eLinkService = "CPXSTREAM";

   /* Input parameter defaults */
   char* target      = "THISCICS";
   char* fpool       = "POOLM2";
   char* cust        = "S000001";
   char* userid      = "VIS01";
   char* password    = "VIS01";
   char* debugQueue  = "        ";

   char* debugSwitch = "";          // Normal Operation
// char* debugSwitch = "offline";   // for testing when eLink not available

/* ------------------------------------------------------------------
If help was requested, display it then exit
------------------------------------------------------------------ */

if (argc > 1) {
     if (strcmp(argv[1], "?")  == 0 ||
       strcmp(argv[1], "-?") == 0 ||
       strcmp(argv[1], "/?") == 0 )
     {
       fprintf(stdout, "Usage:    mwdemo -c customer -t "\
         "target -f pool -u userid -p password\n");
       fprintf(stdout, "Defaults: mwdemo -c %s -t %s -f %s -u %s "\
         "-p %s\n", cust, target, fpool, userid, password);
       exit(1);
     }
} /* end if input parameters are present */

/* ------------------------------------------------------------------
Parse input parameters to override the defaults
------------------------------------------------------------------ */

for (i=1; i < argc; i++) {
   /* If this is a switch and there is a following parameter... */
   if (argv[i][0] == '-' && strlen(argv[i]) == 2 && i != argc-1) {
     c = tolower(argv[i][1]);
     if (c == 't') {           // CrossPlex FEPI Target
       i++;
       target = argv[i];
     } else if (c == 'f') {    // CrossPlex FEPI Pool
```

```
        i++;
        fpool = argv[i];
      } else if (c == 'c') {    // Customer Number
        i++;
        cust = argv[i];
      } else if (c == 'd') {    // Debug Queue Name
        i++;
        debugQueue = argv[i];
      } else if (c == 'u') {    // Userid
        i++;
        userid = argv[i];
      } else if (c == 'p') {    // Password
        i++;
        password = argv[i];
      } else {
        fprintf(stderr, "%s is not a valid switch\n", argv[i]);
        exit(1);
      }
    } else {
      fprintf(stderr, "Input parameter %s is not valid\n", argv[i]);
      exit(1);
    } /* end if this is a switch */
} /* end for */


fprintf(stdout, "Using: mwdemo -c %s -t %s -f %s -u %s -p %s\n",
    cust, target, fpool, userid, password);

/* -----------------------------------------------------------------
Attach to System/T as a Client Process
----------------------------------------------------------------- */

if (strcmp(debugSwitch, "offline") != 0) {
    if (tpinit((TPINIT *) NULL) == -1) {
      fprintf(stderr, "Tpinit failed\n");
      exit(1);
    }
} /* end if not offline testing */

/* -----------------------------------------------------------------
Allocate request & response buffers
----------------------------------------------------------------- */

sbuf = (struct CPXVIEW*) tpalloc("VIEW", "CPXVIEW", sizeof(struct CPXVIEW));
if(sbuf == NULL) {
    fprintf(stderr,"Error allocating send buffer of size %i\n",
      sizeof(struct CPXVIEW));
    tpterm();
    exit(1);
```

```
}

if (sizeof(MWDREC) > sizeof(sbuf->xpmarea)) {
   fprintf(stderr,"sizeof(MWDREC) exceeds sizeof(CPXVIEW->xpmarea)\n");
   tpfree((char *) sbuf);
   tpterm();
   exit(1);
}

rbuf = (struct CPXVIEW*) tpalloc("VIEW", "CPXVIEW", sizeof(struct CPXVIEW));
if(rbuf == NULL) {
   fprintf(stderr,"Error allocating receive buffer of size %i\n", i);
   tpfree((char *) sbuf);
   tpterm();
   exit(1);
}

/* ------------------------------------------------------------------
Setup the inbound Request Buffer
------------------------------------------------------------------ */
memcpy(sbuf, &xpcomm_init, sizeof(struct CPXVIEW)); // Init most fields
memset(sbuf->reserved99, ’ ’, sizeof(sbuf->reserved99));
sbuf->xpalng = sizeof(sbuf->xpmarea);              // Set max size of area
memcpy(sbuf->xpmode, XPMCMDS, sizeof(sbuf->xpmode));// For Command Streams


/* Use the input parameters */
i = strlen(target);
if (i > sizeof(sbuf->xptarg))
   i = sizeof(sbuf->xptarg);
memcpy(sbuf->xptarg, target, i);
for (i = sizeof(sbuf->xptarg) - 1; i >= 0; i--) {
   sbuf->xptarg[i] = toupper(sbuf->xptarg[i]);
}

i = strlen(fpool);
if (i > sizeof(sbuf->xpfpool))
   i = sizeof(sbuf->xpfpool);
memcpy(sbuf->xpfpool, fpool, i);
for (i = sizeof(sbuf->xpfpool) - 1; i >= 0; i--) {
   sbuf->xpfpool[i] = toupper(sbuf->xpfpool[i]);
}

i = strlen(debugQueue);
if (i > sizeof(sbuf->xpdbugq))
   i = sizeof(sbuf->xpdbugq);
memcpy(sbuf->xpdbugq, debugQueue, i);
for (i = sizeof(sbuf->xpdbugq) - 1; i >= 0; i--) {
   sbuf->xpdbugq[i] = toupper(sbuf->xpdbugq[i]);
```

```
}

/* -----------------------------------------------------------------
Execute command stream MWDEMO
----------------------------------------------------------------- */

printf("Calling CrossPlex to execute Command Stream MWDEMO\n");
sbuf->xpilng = sprintf(sbuf->xpmarea, "STREAM(MWDEMO) &USERID(%s) "\
   "&PASSWORD(%s) &CUST(%s)", userid, password, cust);
if (strcmp(debugSwitch, "offline") != 0) {
   ret = tpcall(eLinkService, (char *) sbuf, 0, (char**) &rbuf,
        &rcvlen, (long)0);
   if(ret == -1) {
     fprintf(stderr, "Can’t send request to service %s\n",
       eLinkService);

     fprintf(stderr, "Tperrno = %d\n", tperrno);
     tpfree((char *) sbuf);
     tpfree((char *) rbuf);
     tpterm();
     exit(1);
   }
}

/* -----------------------------------------------------------------
Check for and Handle CrossPlex-detected errors
----------------------------------------------------------------- */

i = cpxError(rbuf);
if (i) {
   tpfree((char *) sbuf);
   tpfree((char *) rbuf);
   tpterm();
   exit(1);
} /* end if CrossPlex errors */

/* -----------------------------------------------------------------
Display the results
----------------------------------------------------------------- */

printf("Returned data:\n\n");

mwdrec = (MWDREC *) &rbuf->xpmarea;  // Address the returned record
printf("Consolidated Customer Inquiry\n\n");
printf("Customer Number:  %.7s\n",  mwdrec->custno);
printf("Customer Name:    %.25s\n", mwdrec->name);
printf("Street address:   %.25s\n", mwdrec->addr1);
printf("Mail address:     %.25s\n", mwdrec->addr2);
/* Trim trailing blanks from the name of the City */
```

```
for (i=sizeof(mwdrec->city)-1; i >= 0 && mwdrec->city[i]==’ ’;i--) {
   mwdrec->city[i] = ’\0’;
}
printf("City, State Zip:  %.25s, %.2s %.5s\n", mwdrec->city,
   mwdrec->state, mwdrec->zip);
printf("Home phone:       %.3s-%.4s\n", mwdrec->phone,
     &mwdrec->phone[3]);
printf("Business phone:   %.3s-%.4s\n", mwdrec->busphone,
     &mwdrec->busphone[3]);
printf("Start date:       %.2s/%.2s/%.2s\n", mwdrec->sdate,
     &mwdrec->sdate[2], &mwdrec->sdate[4]);
printf("Initial Deposit:  $ %.4s.%.2s\n", mwdrec->deposit,
     &mwdrec->deposit[4]);
printf("Total Amount due: $ %.4s.%.2s\n", mwdrec->totdue,
     &mwdrec->totdue[4]);
printf("\n\nPayment history for previous six months:\n");
printf("\nMonth  Amount   Date      Amount due\n");
printf("-----  -------  -------   ----------\n");
for (i=0; i < sizeof(mwdrec->payhist)/sizeof(MWDRECPAYHIST); i++) {
   printf(" %.3s   %.4s.%.2s  %.2s/%.2s/%.2s  $ %.5s.%.2s\n",
      mwdrec->payhist[i].pmonth,
      mwdrec->payhist[i].pamount, &mwdrec->payhist[i].pamount[4],
      mwdrec->payhist[i].pdate, &mwdrec->payhist[i].pdate[2],
      &mwdrec->payhist[i].pdate[4], mwdrec->payhist[i].pamtdue,
      &mwdrec->payhist[i].pamtdue[5]);
} /* end for loop */

/* -----------------------------------------------------------------
Free Buffers & Detach from Tuxedo
----------------------------------------------------------------- */

tpfree((char *) sbuf);
tpfree((char *) rbuf);
tpterm();
} /* end main() */
```

Task 4: Configure eAM to talk to the CrossPlex installation on your mainframe.

BEA eLink Adapter for Mainframe must be configured to communicate with the CICS region that hosts your CrossPlex installation. Refer to the "Configuring the System" section for details of configuring eAM. Listing 7-4 is a fragment of an eAM `DMCONFIG` file that illustrates the remote service entries that are needed for this example. These entries must be modified to reference the actual local and remote domains in your configuration.

Listing 7-4 Sample DMCONFIG File Fragment

```
#------------------
*DM_REMOTE_SERVICES
CPXSTREAM  AUTOTRAN=N
           LDOM=LOCAL1
           RDOM=REMOTE1
           CONV=N
           RNAME="CPXSTREAM"
           INBUFTYPE="VIEW:CPXVIEW"
           OUTBUFTYPE="VIEW:CPXVIEW"

CPXTEXT    AUTOTRAN=N
           LDOM=LOCAL1
           RDOM=REMOTE1
           CONV=N
           RNAME="CPXTEXT"
           INBUFTYPE="STRING"
           OUTBUFTYPE="STRING"
```

Task 5: Use the Application

To use the application, complete the following steps:

Step 1: Start Your ATMI System.

Use the `tmboot` command to start the ATMI application containing the eAM gateway. Verify that the gateway starts successfully and that connection to your mainframe is established.

Step 2: Run the Client Program.

The syntax for invocation of the sample client program is:

```
mwdemo {options..}
```

In this argument, options may be defined in one or more of the following ways:

`-c customer`
: Specifies the number of the customer to access from the mainframe application.

`-t target`
: Specifies the CrossPlex FEPI target to access.

`-f pool`
: Specifies the CrossPlex FEPI pool to access.

`-u userid`
: Specifies the user ID to use for mainframe access.

`-p password`
: Specifies the password to use for mainframe access.

When the program runs successfully, it will print a report similar to the following example:

```
Consolidated Customer Inquiry

Customer Number:   1234567
Customer Name:     Acme
Street Address:    123 Main St.
Main Address:      PO Box 500
City, State Zip:   Plano, TX 75093
Home Phone:        555-1234
Business Phone:    555-4321
Start date:        08/12/61
```

Initial Deposit:	$ 5200.00
Total Amout due:	$ 125.00

Payment history for previous six months:

Month	Amount	Date	Amount due
Jan	100.00	01/12/00	$ 625.00
Feb	100.00	02/10/00	$ 525.00
Mar	100.00	03/14/00	$ 425.00
Apr	100.00	04/12/00	$ 325.00
May	100.00	05/12/00	$ 225.00
Jun	100.00	06/13/00	$ 125.00

A Reference Pages

This section covers the following reference pages, formerly called man pages:

- `addumap`
- `addusr`
- `CRMLOGS`
- `crmlkoff`
- `crmlkon`
- `delumap`
- `delusr`
- `DMADM`
- `dmadmin`
- `dmconfig`
- `dmloadcf`
- `dmunloadcf`
- `GWADM`
- `GWSNAX`
- `modusr`
- `SNACRM`
- `xsnacrm`

addumap

Adds a local-to-remote mapping for a local/remote domain pair.

Synopsis

```
addumap -d <local domain ID> -R <remote domain ID>
-p <local principal name> -u <remote username>
```

Description

`addumap` can only be executed as a subcommand of `dmadmin`(1). The purpose of this page is to describe options for the subcommand and to show examples.

The subcommand allows the administrator to add local-to-remote user mappings for a local/remote domain pair.

Mappings are defined to be inbound, outbound or both when the application is using SNA-type gateways and `SECURITY` is set to `USER_AUTH`, `ACL`, or `MANDATORY ACL` in the `ubbconfig` file and `SECURITY` is set to `DM_PW` or `USER_PW` in the `DMCONFIG` file.

The following options are available:

`-d <`*`local domain ID`*`>`
: This is the name of the local domain gateway with which the ids and passwords are associated. This is the same ID as the one used when creating the domain definitions either in the `DMCONFIG` file or through the Graphical Administrative Interface.

`-R <`*`remote domain ID`*`>`
: This is the name of the remote domain gateway with which the ids and passwords are associated. This is the same ID as the one used when creating the domain definitions either in the `DMCONFIG` file or through the Graphical Administrative Interface.

`-p` <*`local principal`*>
: The user identification number. The *`local principal`* must be defined in the ACL user file and must be unique within the list of existing identifiers for the application.

`-u` <*`remote username`*>
: The remote user name as defined in the ACL security application (for example, RACF) of the remote domain.

Before running this subcommand the application must be configured using either the Graphical Administrative Interface or `tmloadcf`(1) and `dmloadcf`(1). `dmadmin addumap` may be run on any active node.

Portability

This subcommand is available on the latest version of Tuxedo, as documented for this release of BEA eLink Adapter for Mainframe.

Diagnostics

The `dmadmin addumap` subcommand exits with a return code of 0 upon successful completion.

Example

```
addumap -d ldom -R cdom -p tuxusr -u CICSUSR
            /*maps principal tuxusr with
            remote user cicsusr */
```

See Also

`dmadmin`(1), `delumap`(5)

addusr

Adds a user to the remote domain user and password file.

Synopsis

```
addusr -d <local domain ID> -R <remote domain ID> -u <remote
username>
[-w ]
```

Description

`addusr` can only be executed as a subcommand of `dmadmin`(1). The purpose of this page is to describe options for the subcommand and to show an example.

The subcommand allows the administrator to add remote user names and passwords to the remote domain remote user and password table. If `-w` is not specified, the user is prompted for a password.

The table entries created are used for passing remote user names and passwords to remote SNA domains when the application is using SNA-type gateways and `SECURITY` is set to `USER_AUTH`, `ACL`, or `MANDATORY ACL` in the `ubbconfig` file and `SECURITY` is set to `DM_PW` or `USER_PW` in the `DMCONFIG` file.

The following options are available:

`-d` <*`local domain ID`*>
: This is the name of the local domain gateway with which the ids and passwords are associated. This is the same ID as the one used when creating the domain definitions either in the `DMCONFIG` file or through the Graphical Administrative Interface.

`-R` <*`remote domain ID`*>
: This is the name of the remote domain gateway with which the ids and passwords are associated. This is the same ID as the one used when creating

the domain definitions either in the `DMCONFIG` file or through the Graphical Administrative Interface.

`-u` *`<remote username`*`>`
: The remote user name to be added.

`-w`
: Do not prompt for password.

Before running this subcommand the application must be configured using either the Graphical Administrative Interface or `tmloadcf`(1) and `dmloadcf`(1). `dmadmin addusr` may be run on any active node.

Portability

This subcommand is available on the latest version of Tuxedo, as documented for this release of BEA eLink Adapter for Mainframe.

Diagnostics

The `dmadmin addusr` subcommand exits with a return code of 0 upon successful completion.

Examples

```
addusr -d tux -R cics -u CICSUSR   /*adds remote user CICSUSR to
                                     cics domain’s user and
                                     password file. The
                                     administrator is prompted for
                                    a password*/
```

See Also

`delusr`(5), `modusr`(5)

CRMLOGS

Displays the content and state of the Communications Resource Manager (`CRM`) log files.

Synopsis

```
CRMLOGS <group> [<crm name>]
```

Description

You can use the `CRMLOGS` command to display the contents and state of the two `SNARCM` log files. `RSTRTLOG` is the transaction state log used during the recovery process and the `BLOBLOG` log stores session and link information. Deleting the log files require a cold start for each link involved.

`CRMLOGS` requires the following parameters:

`group`
: SNA domain group name (required)

`crm name`
: `CRM` name (default SNACRM)

Diagnostics

`CRMLOGS` exits with a return code of 0 upon successful completion.

Examples

To display the `RSTRTLOG` log file for group2, type:

```
CRMLOGS GROUP2 SNARCM.GROUP2.RSTRTLOG
```

To display the `BLOBLOG` log file for group1, type:

```
CRMLOGS GROUP1 SNARCM.GROUP1.BLOBLOG
```

See Also

SNACRM and `xsnacrm`

crmlkoff

Stops one or more named CRM links.

Synopsis

```
crmlkoff -n<hostname:port> [-v -i -h -u<keyfile>] <linkname> ...
```

Description

`crmlkoff` stops all of the CRM links named on the command line. This is useful if one or more individual links need to be stopped after the CRM server booted. It can be used from any machine located on the same TCP/IP network as the machine running the CRM server. It can be used in a script and will return zero if the command could be sent to the target CRM. It will return one if the command could not be sent to the target CRM.

`-n`
: Names the machine and port running the CRM server.

`-v`
: Specifies verbose. Normally the command will not produce any messages, facilitating use in a script.

`-i`
: Ignores errors. When specifying multiple links, any error encountered when issuing CRM commands will cause crmlkoff to stop processing links and return. Errors can be ignored for individual links, and processing continued with the next named link.

`-u<keyfile>`
: Establishes that process authentication is in effect for communications between this process and the CRM.
`<keyfile>` is the location and the file containing a hash key known to both this process and the CRM. The file contains a single line specifying a unique hash key (limited to eight characters). The file should be protected.

Note: If the CRM is running under MVS, the `-u` option should be specified as: `-u DD:ddname.`

In this argument, `ddname` is a 1 to 8 byte `DD` statement that will identify the dataset name in the JCL.

`<linkname>`
Names the link to be stopped. This is the *DM_SNALINKS entry in the DMCONFIG which defines this link. Multiple link names can be specified.

Portability

`crmlkoff` is supported as a Tuxedo-supplied administrative tool on all platforms supporting an eAM CRM.

Example

To stop links `link1` and `cicstest` owned by the CRM running on `mach` at port 5000:

```
crmlkoff -n mach:5000 link1 cicstest
```

Diagnostics

`crmlkoff` only checks the syntax of the command. Separate facilities, either xsnacrm or mainframe based facilities must be used to determine if the link actually became inactive. If the command could not be successfully sent to the CRM `crmlkoff` prints an error message, if in verbose mode, and exits with error code 1. Upon successful completion, `crmlkoff` exits with exit code 0.

See Also

`crmlkon(1), xsnacrm(1)`

ATMI platform User Guide

crmlkon

Starts one or more named CRM links.

Synopsis

```
crmlkon -n<hostname:port> [-v -i -h -u<keyfile>] <linkname> ...
```

Description

`crmlkon` starts all of the CRM links named on the command line. This is useful if one or more individual links failed to start when the CRM server booted. It can be used from any machine located on the same TCP/IP network as the machine running the CRM server. It can be used in a script and will return zero if the command could be sent to the target CRM. It will return one if the command could not be sent to the target CRM.

`-n`

Names the machine and port running the CRM server.

`-v`

Specifies verbose. Normally the command will not produce any messages, facilitating use in a script.

`-i`

Ignores errors. When specifying multiple links, any error encountered when issuing CRM commands will cause crmlkon to stop processing links and return. Errors can be ignored for individual links, and processing continued with the next named link.

`-u<keyfile>`

Establishes that process authentication is in effect for communications between this process and the CRM.
`<keyfile>` is the location and the file containing a hash key known to both this process and the CRM. The file contains a single line specifying a unique hash key (limited to eight characters). The file should be protected.

Note: If the CRM is running under MVS, the `-u` option should be specified as: `-u DD:ddname.`

In this argument, `ddname` is a 1 to 8 byte `DD` statement that will identify the dataset name in the JCL.

`<linkname>`

Names the link to be started. This is the *DM_SNALINKS entry in the DMCONFIG which defines this link. Multiple link names can be specified.

Portability

`crmlkon` is supported as a Tuxedo-supplied administrative tool on all platforms supporting an eLink CRM.

Example

To start links `link2` and `cicstest` owned by the CRM running on `mach1` at port 5000:

```
crmlkon -n mach1:5000 link2 cicstest
```

Diagnostics

`crmlkon` only checks the syntax of the command. Separate facilities, either xsnacrm or mainframe based facilities must be used to determine if the link actually became active. It the command could not be successfully sent to the CRM `crmlkon` prints an error message, if in verbose mode, and exits with error code 1. Upon successful completion, `crmlkon` exits with exit code 0.

See Also

`crmlkon(1), xsnacrm(1)`

delumap

Deletes a local-to-remote mapping for a local/remote domain pair.

Synopsis

```
delumap -d <local domain ID> -R <remote domain ID>
-p <local principal name> -u <remote username>
```

Description

`delumap` can only be executed as a subcommand of `dmadmin`(1). The purpose of this page is to describe options for the subcommand and to show examples.

The subcommand allows the administrator to delete local-to-remote user mappings for a local/remote domain pair.

Mappings are defined to be inbound, outbound or both when the application is using SNA-type gateways and `SECURITY` is set to `USER_AUTH`, `ACL`, or `MANDATORY ACL` in the `ubbconfig` file and `SECURITY` is set to `DM_PW` or `USER_PW` in the `DMCONFIG` file.

The following options are available:

`-d` *l<ocal domain ID>*
: This is the name of the local domain gateway with which the ids and passwords are associated. This is the same ID as the one used when creating the domain definitions either in the `DMCONFIG` file or through the Graphical Administrative Interface.

`-R` *<remote domain ID>*
: This is the name of the remote domain gateway with which the ids and passwords are associated. This is the same ID as the one used when creating the domain definitions either in the `DMCONFIG` file or through the Graphical Administrative Interface.

`-p <`*`local principal`*`>`

The user identification number. The *`local principal`* must be defined in the ACL user file and must be unique within the list of existing identifiers for the application.

`-u <`*`remote username`*`>`

The remote user name as defined in the ACL security application (for example, RACF) of the remote domain. Space is a valid remote username.

Before running this subcommand the application must be configured using either the Graphical Administrative Interface or `tmloadcf`(1) and `dmloadcf`(1). `dmadmin delumap` may be run on any active node.

Portability

This subcommand is available on the latest version of Tuxedo, as documented for this release of BEA eLink Adapter for Mainframe.

Diagnostics

The `dmadmin delumap` subcommand exits with a return code of 0 upon successful completion.

Example

```
delumap -d ldom -R cics -p tuxusr -u CICSUSR
                /*deletes the mapping of principal
                 tuxusr with remote user cicsusr */
```

See Also

`dmadmin`(1), `addumap`(5)

delusr

Deletes a user from the remote domain user and password file.

Synopsis

```
delusr -d <local domain> -R <remote domain> -u <remote username>
```

Description

`delusr` can only be executed as a subcommand of `dmadmin`(1). The purpose of this page is to describe options for the subcommand and to show an example.

The subcommand allows the administrator to remove remote user names and passwords from the remote domain remote user and password table.

Once the entries are deleted they can no longer be used for mapping remote user names and passwords to local user names and passwords when the application is using SNA-type gateways and `SECURITY` is set to `USER_AUTH`, `ACL`, or `MANDATORY ACL` in the `ubbconfig` file and `SECURITY` is set to `DM_USER_PW` in the `DMCONFIG` file.

The following options are available:

`-d <`*`local domain ID`*`>`
: This is the name of the local domain gateway with which the ids and passwords are associated. This is the same ID as the one used when creating the domain definitions either in the `DMCONFIG` file or through the Graphical Administrative Interface.

`-R <`*`remote domain ID`*`>`
: This is the name of the remote domain gateway with which the ids and passwords are associated. This is the same ID as the one used when creating the domain definitions either in the `DMCONFIG` file or through the Graphical Administrative Interface.

-u <*remote username*>

The remote user name to be deleted.

Before running this subcommand the application must be configured using either the Graphical Administrative Interface or `tmloadcf`(1) and `dmloadcf`(1). `dmadmin delusr` may be run on any active node.

Portability

This subcommand is available on the latest version of Tuxedo, as documented for this release of BEA eLink Adapter for Mainframe.

Diagnostics

The `dmadmin delusr` subcommand exits with a return code of 0 upon successful completion.

Examples

```
delusr -d tux -R cics -u CICSUSR  /*deletes remote user CICSUSR to
                                   cics domain users. The
                                   administrator is prompted for a
                                   password*/
```

See Also

`addusr`(5), `modusr`(5)

DMADM

/Domain administrative server.

Synopsis

```
DMADM SRVGRP = “identifier”
   SRVID = “number”
   REPLYQ = “N”
```

Description

The /DOMAIN administrative server (`DMADM`) is a Tuxedo-supplied server that provides run-time access to the binary domain configuration file (`BDMCONFIG` file). When `DMADM` is booted, the `BDMCONFIG` environment variable should be set to the pathname of the file containing the binary version of the `DMCONFIG` file.

`DMADM` is described in the SERVERS section of the `UBBCONFIG` file as a server running within a group, e.g., `DMADMGRP`. There should be only one instance of the `DMADM` running in this group and it must not have a reply queue (`REPLYQ` must be set to “N”).

The following server parameters can also be specified for the `DMADM` server in the SERVERS section: `SEQUENCE`, `ENVFILE`, `MAXGEN`, `GRACE`, `RESTART`, `RQPERM` and `SYSTEM_ACCESS`.

Portability

`DMADM` is supported as a Tuxedo-supplied server on UNIX System and Windows NT operating systems.

Examples

The following example illustrates the definition of the administrative server and a gateway group in the UBBCONFIG file.

```
#
*GROUPS
DMADMGRP  LMID=mach1 GRPNO=1
gwgrp     LMID=mach1 GRPNO=2
#
*SERVERS
DMADM SRVGRP="DMADMGRP" SRVID=1001 REPLYQ=N RESTART=Y GRACE=0
GWADM SRVGRP="gwgrp" SRVID=1002 REPLYQ=N RESTART=Y GRACE=0
GWSNAX SRVGRP="gwgrp" SRVID=1003 RQADDR="gwgrp" REPLYQ=N
RESTART=N MIN=1 MAX=1
```

See Also

dmadmin(1), tmboot(1), dmconfig(5), GWADM(5), servopts(5), ubbconfig(5)

Tuxedo /Domain User Guide
Tuxedo Administrator's Guide

dmadmin

Tuxedo System/T Domain Administration Command Interpreter.

Synopsis

```
dmadmin [-c]
```

Description

The `dmadmin` interactive command interpreter is used for the administration of domain gateway groups defined for a particular Tuxedo System/T application. The interpreter can operate in two modes: administration mode and configuration mode.

The `dmadmin` command interpreter enters *administration* mode when called with no parameters. This is the default. In this mode, `dmadmin` can be run on any active node (excluding workstations) within an active application. Application administrators can use this mode to obtain or change parameters on any active domain gateway group. Application administrators may also use this mode to create, destroy, or re-initialize the `DMTLOG` for a particular local domain. In this case, the domain gateway group associated with that local domain must not be active, and `dmadmin` must be run on the machine assigned to the corresponding gateway group.

The `dmadmin` command interpreter enters *configuration* mode when it is invoked with the `-c` option or when the `config` subcommand is invoked. Application administrators can use this mode to update or add new configuration information to the binary version of the domain configuration file (`BDMCONFIG`).

The `dmadmin` command interpreter requires the use of the `DOMAIN` administrative server (`DMADM`) for the administration of the `BDMCONFIG` file and the gateway administrative server (`GWADM`) for the re-configuration of active `DOMAIN` gateway groups (there is one `GWADM` per gateway group).

Administration Mode Commands

Once `dmadmin` has been invoked, commands may be entered at the prompt (“>”) according to the following syntax:

```
command [arguments]
```

Several commonly occurring arguments can be given default values using the default command. Commands that accept parameters set using the default command. Check default to see if a value has been set. If no value is set, an error message is returned.

Once set, a default value remains in effect until the session is ended, unless changed by another default command. Defaults may be overridden by entering an explicit value on the command line, or reset by entering the value “*”. The effect of an override lasts for a single instance of the command.

Output from `dmadmin` commands is paginated according to the pagination command in use (see the paginate subcommand below).

Commands may be entered either by their full name or their abbreviation (shown in parentheses) followed by any appropriate arguments. Arguments appearing in square brackets, [ ], are optional; those in curly braces, `{}`, indicate a selection from mutually exclusive options. Note that for many commands *`local_domain_name`* is a required argument, but commands can be set with the default command.

The following commands are available in administration mode:

`addumap [ options ]`
: Add local user mappings to remote user mappings for a local/remote domain pair. Mappings are defined to be inbound, outbound or both. See the `addumap`(5) reference page for an explanation of the available options and for examples.

`addusr (addu) [ options ]`
: Add remote user names and passwords to the remote user and password tables of a remote domain. See the `addusr`(5) reference page for an explanation of the available options and for examples.

`advertise (adv) –d` *`local_domain_name`* `[{ –all | `*`service`*`}]`
: Advertise all remote services provided by the named local domain or the specified remote service.

`audit (audit) –d` *`local_domain_name`* `[{off | on}]`

Activate (on) or deactivate (off) the audit trace for the named local domain. If no option is given, then the current setting will be toggled between the values on and `off`, and the new setting will be printed. The initial setting is `off`.

`chbktime (chbt) –d` *`local_domain_name`* `–t` *`bktime`*

Change the blocking timeout for a particular local domain.

`config (config)`

Enter configuration mode. Commands issued in this mode follow the conventions defined in the section "Configuration Mode Commands" (see below).

`crdmlog (crdlg) –d` *`local_domain_name`*

Create the domain transaction log for the named local domain on the current machine (that is, the machine where `dmadmin` is running). The command uses the parameters specified in the `DMCONFIG` file. This command fails if the named local domain is active on the current machine or if the log already exists.

`default (d) [–d` *`local_domain_name`*`]`

Set the corresponding argument to be the default local domain. Defaults may be reset by specifying "*" as an argument.

If the `default` command is entered with no arguments, the current default values are printed.

`delumap [ options ]`

Delete local to remote user mappings for a local/remote domain pair. See the `delumap`(5) reference page for an explanation of the available options and for examples.

`delusr (delu) [ options ]`

Delete remote user names and passwords from the remote user and password tables of a remote domain. See the `delusr`(5) reference page for an explanation of the available options and for examples.

`dsdmlog (dsdlg) –d` *`local_domain_name`* `[ –y ]`

Destroy the domain transaction log for the named local domain on the current machine (that is, the machine where `dmadmin` is running). An error is returned if a `DMTLOG` is not defined for this local domain, if the local domain is active, or if outstanding transaction records exist in the log. The term outstanding transactions means that a global transaction has been committed but an end-of-transaction has not yet been written. This command prompts for confirmation before proceeding unless the `-y` option is specified. `dsdmlog` is not supported for SNA-type gateways.

`echo (e) [{off | on}]`
Echo input command lines when set to `on`. If no option is given, then the current setting is toggled, and the new setting is printed. The initial setting is `off`.

`forgettrans (ft) –d` *`local_domain_name`* `[ –t` *`tran_id`*`]`
Forget one or all heuristic log records for the named local domain. If the transaction identifier *`tran_id`* is specified, then only the heuristic log record for that transaction will be forgotten. The transaction identifier *`tran_id`* can be obtained from the `printtrans` command or from the `ULOG` file. `forgettrans` is not supported for SNA-type gateways.

`help (h) [`*`command`*`]`
Print help messages. If *`command`* is specified, the abbreviation, arguments, and description for that command are printed. Omitting all arguments causes the syntax of all commands to be displayed.

`indmlog (indlg) –d` *`local_domain_name`* `[ –y ]`
Re-initialize the domain transaction log for the named local domain on the current machine (that is, the machine where `dmadmin` is running). An error is returned if a `DMTLOG` is not defined for this local domain, if the local domain is active, or if outstanding transaction records exist in the log. The term outstanding transactions means that a global transaction has been committed but an end-of-transaction has not yet been written. The command prompts for confirmation before proceeding unless the `-y` option is specified. `indmlog` is not supported for SNA-type gateways.

`modusr (modu) [ options ]`
Change remote passwords in the password tables of a remote domain. See the `modusr`(5) reference page for an explanation of the available options and for examples.

`paginate (page) [{off | on}]`
Paginate output. If no option is given, then the current setting will be toggled, and the new setting is printed. The initial setting is on, unless either standard input or standard output is a non-tty device. Pagination may only be turned on when both standard input and standard output are tty devices. The shell environment variable PAGER may be used to override the default command used for paging output. The default paging command is the indigenous one to

the native operating system environment, for example, the command `pg` is the default on UNIX System operating environments.

`passwd (passwd) [ -r ]` *`local_domain_name remote_domain_name`*
: Prompts the administrator for new passwords for the specified local and remote domains. The `-r` option specifies that existing passwords and new passwords should be encrypted using a new key generated by the system. The password is truncated after at most eight characters.

`printdomain (pd) –d` *`local_domain_name`*
: Print information about the named local domain. Information printed includes connected remote domains, global information shared by the gateway processes, and additional information that is dependent on the domain type instantiation.

`printstats (stats) –d` *`local_domain_name`*
: Print statistical and performance information gathered by the named local domain. The information printed is dependent on the domain gateway type.

`printtrans (pt) –d` *`local_domain_name`*
: Print transaction information for the named local domain. `printtrans` is not supported for SNA-type gateways.

`quit (q)`
: Terminate the session.

`resume (res) –d` *`local_domain_name`* `[{ –all  |` *`service`*`}]`
: Resume processing of the specified service or for all remote services handled by the named local domain.

`stats (stats) –d` *`local_domain_name`* `[{ off | on | reset }]`
: Activate (*`on`*), deactivate (*`off`*), or reset (*`reset`*) statistics gathering for the named local domain. If no option is given, then the current setting will be toggled between the values *`on`* and *`off`*, and the new setting will be printed. The initial setting is `off`.

`suspend (susp) –d` *`local_domain_name`* `[{ –all  |` *`service`*`}]`
: Suspend one or all remote services for the named local domain.

`unadvertise (unadv) –d` *`local_domain_name`* `[{ –all  |` *`service`*`}]`
: Unadvertise one or all remote services for the named local domain.

`verbose (v) [{off | on}]`

Produce output in verbose mode. If no option is given, then the current setting will be toggled, and the new setting is printed. The initial setting is `off`.

`! *shellcommand*`

Escape to shell and execute *`shellcommand`*.

`!!`

Repeat previous shell command.

`# [*text*]`

Lines beginning with "#" are comment lines and are ignored.

`<CR>`

Repeat the last command.

Configuration Mode Commands

The `dmadmin` command enters configuration mode when executed with the `-c` option or when the `config` subcommand is used. In this mode, `dmadmin` allows run-time updates to the `BDMCONFIG` file. `dmadmin` manages a buffer that contains input field values to be added or retrieved, and displays output field values and status after each operation completes. The user can update the input buffer using any available text editor.

`The dmadmin` command first prompts for the desired section followed by a prompt for the desired operation.

The prompt for the section is as follows:

```
Sections:
       1) LOCAL_DOMAINS     2) REMOTE_DOMAINS
       3) LOCAL_SERVICES    4) REMOTE_SERVICES
       5) ROUTING           6) ACCESS_CONTROL
       7) PASSWORDS         8) TDOMAIN
       9) OSITP            10) SNA
      11) QUIT
Enter Section [1]:
```

The number of the default section appears in square brackets at the end of the prompt. You can accept the default by pressing `RETURN` or `ENTER`. To select another section enter its number, then press `RETURN` or `ENTER`.

`dmadmin` then prompts for the desired operation.

```
Operations:
       1) FIRST              2) NEXT
       3) RETRIEVE           4) ADD
       5) UPDATE             6) DELETE
       7) NEW_SECTION        8) QUIT
Enter Operation [1]:
```

The number of the default operation is printed in square brackets at the end of the prompt. Pressing RETURN or ENTER selects this option. To select another operation enter its number, then press RETURN or ENTER.

The currently supported operations are:

1. FIRST

   Retrieve the first record from the specified section. No key fields are needed (they are ignored if in the input buffer).

2. NEXT

   Retrieve the next record from the specified section, based on the key fields in the input buffer.

3. RETRIEVE

   Retrieve the indicated record from the specified section by key field(s) (see fields description below).

4. ADD

   Add the indicated record in the specified section. Any fields not specified (unless required) take their default values as specified in `dmconfig`(5). The current value for all fields is returned in the output buffer. This operation can only be done by the System/T administrator.

5. UPDATE

   Update the record specified in the input buffer in the selected section. Any fields not specified in the input buffer remain unchanged. The current value for all fields is returned in the input buffer. This operation can only be done by the System/T administrator.

6. DELETE

   Delete the record specified in the input buffer from the selected section. This operation can only be done by the System/T administrator.

7. NEW SECTION
   Clear the input buffer (all fields are deleted). After this operation, `dmadmin` immediately prompts for the section again.

8. QUIT
   Exit the program gracefully (`dmadmin` is terminated). A value of `q` for any prompt also exits the program.

For configuration operations, the effective user identifier must match the System/T administrator user identifier (`UID`) for the machine on which this program is executed. When a record is updated or added, all default values and validations used by `dmloadcf`(1) are enforced.

`dmadmin` then prompts whether or not to edit the input buffer.

```
Enter editor to add/modify fields [n]?
```

Entering a value of `y` will put the input buffer into a temporary file and execute the text editor. The environment variable `EDITOR` is used to determine which editor to be used; the default is "`ed`". The input format is in field name/field value pairs and is described in the CONFIGURATION INPUT FORMAT section below. The field names associated with each `DMCONFIG` section are listed in tables in the subsections below. The semantics of the fields and associated ranges, default values, restrictions, etc., are described in `dmconfig`(5). In most cases, the field name is the same as the `KEYWORD` in the `DMCONFIG` file, prefixed with "TA_". When the user completes editing the input buffer, `dmadmin` reads it. If more than one line occurs for a particular field name, the first occurrence is used and other occurrences are ignored. If any errors occur, a syntax error will be printed and dmadmin prompts whether or not to correct the problem.

```
Enter editor to correct?
```

If the problem is not corrected (response `n`), then the input buffer will contain no fields. Otherwise, the editor is executed again.

Finally, `dmadmin` asks if the operation should be done.

```
Perform operation [y]?
```

When the operation completes, `dmadmin` prints the return value as in

```
Return value TAOK
```

followed by the output buffer fields. The process then begins again with a prompt for the section. All output buffer fields are available in the input buffer unless the buffer is cleared.

Entering break at any time restarts the interaction at the prompt for the section.

When "`QUIT`" is selected, `dmadmin` prompts for authorization to create a backup ASCII version of the configuration:

```
Unload BDMCONFIG file into ASCII backup [y]?
```

If a backup is selected, `dmadmin` prompts for the file name.

```
Backup filename [DMCONFIG]?
```

On success, `dmadmin` indicates that a backup was created, otherwise an error is printed.

Configuration Input Format

Input packets consist of lines formatted as follows:

```
fldname<tabs>fldval
```

The field name is separated from the field value by one or more tabs (or spaces).

Lengthy field values can be continued on the next line by having the continuation line begin with one or more tabs (which are dropped when read back into `dmadmin`).

Empty lines consisting of a single newline character are ignored.

To enter an unprintable character in the field value or to start a field value with a tab, use a backslash followed by the two-character hexadecimal representation of the desired character (see ASCII(5) in a UNIX reference manual). A space, for example, can be entered in the input data as \20. A backslash can be entered using two backslash characters. `dmadmin` recognizes all input in this format, but its greatest usefulness is for non-printing characters.

Configuration Limitations

The following are general limitations of the dynamic domain re-configuration capability:

- Values for key fields (as indicated in the following sections) may not be modified. Key fields can be modified, when the system is down, by reloading the configuration file.

- Dynamic deletions cannot be applied when local domains are active (the corresponding gateway group is running).

Restrictions for Configuration Field Identifiers/Updates

The following sections describe the following information for each `DMCONFIG` section:

- Field identifiers for each `DMCONFIG` field
- Field type of identifier
- Field updates

All applicable field values are returned with the retrieval operations. Fields that are allowed and/or required for adding a record are described in `dmconfig`(5). Fields indicated below as *`key`* are key fields that are used to uniquely identify a record within section. These key fields are required to be in the input buffer when updates are done and are not allowed to be updated dynamically. The `Update` column indicates when a field can be updated. The possible values are:

Yes
: Can be updated at any time.

NoGW
: Cannot be updated dynamically while the gateway group representing the local domain is running.

No
: Cannot be updated dynamically while at least one gateway group is running.

Configuring the DM_LOCAL_DOMAINS Section

The following table lists the fields in the DM_LOCAL_DOMAINS section.

Table A-1 DM_LOCAL_DOMAINS SECTION

Field Identifier	Field Type	Update	Notes
TA_LDOM	string	NoGW	key

Table A-1 DM_LOCAL_DOMAINS SECTION

Field Identifier	Field Type	Update	Notes
TA_AUDITLOG	string	Yes	
TA_BLOCKTIME	numeric	Yes	
TA_DOMAINID	string	NoGW	
TA_DMTLOGDEV	string	NoGW	
TA_DMTLOGNAME	string	NoGW	
TA_DMTLOGSIZE	numeric	NoGW	
TA_GWGRP	string	NoGW	
TA_MAXDATALEN	numeric	Yes	
TA_MAXRDOM	numeric	Yes	
TA_MAXRDTRAN	numeric	NoGW	
TA_MAXTRAN	numeric	NoGW	
TA_SECURITY	string	Yes	format: {NONE | APP_PW | DM_PW}
TA_TYPE	string	NoGW	format: {TDOMAIN | OSITP | SNA}

Configuring the DM_REMOTE_DOMAINS Section

The following table lists the fields in the DM_REMOTE_DOMAINS section.

Table A-2 DM_REMOTE_DOMAINS SECTION

Field Identifier	Field Type	Update	Notes
TA_RDOM	string	No	key
TA_DOMAINID	string	No	
TA_TYPE	string	No	format: {TDOMAIN | OSITP | SNA}

Table A-2 DM_REMOTE_DOMAINS SECTION

Field Identifier	Field Type	Update	Notes
TA_CODEPAGE	string	No	CODEPAGE filename

Configuring the DM_TDOMAIN Section

The DM_TDOMAIN section contains the network addressing parameters required by `TDOMAIN` type domains. The following lists the fields in this section:

Table A-3 DM_TDOMAIN SECTION

Field Identifier	Field Type	Update	Notes
TA_LDOM or TA_RDOM	string	No/NoGW	key
TA_NWADDR	string	No/NoGW	ASCII format (no embedded NULL characters)

If the domain identifier (`TA_LDOM`) is a local domain identifier, then the `TA_NWADDR` field can be updated if the gateway group representing that local domain is not running.

Configuring the DM_OSITP Section

The DM_OSITP section contains the network addressing parameters required by `OSITP` type domains. The following lists the fields in this section:

Table A-4 DM_OSITP SECTION

Field Identifier	Field Type	Update	Notes
TA_LDOM or TA_RDOM	string	No/NoGW	key
TA_APT	string	No/NoGW	

Table A-4 DM_OSITP SECTION

Field Identifier	Field Type	Update	Notes
TA_AEQ	string	No/NoGW	
TA_AET	string	No/NoGW	
TA_ACN	string	No/NoGW	
TA_APID	string	No/NoGW	
TA_AEID	string	No/NoGW	
TA_PROFILE	string	No/NoGW	

If the domain identifier (`TA_LDOM`) is a local domain identifier, then the other fields in this table can be updated if the gateway group representing that local domain is not running.

Configuring the DM_LOCAL_SERVICES Section

The following table lists the fields in the DM_LOCAL_SERVICES section.

Table A-5 DM_LOCAL_SERVICES SECTION

Field Identifier	Field Type	Update	Notes
TA_SERVICENAME	string	No	key
TA_LDOM	string	Yes	
TA_RNAME	string	Yes	
TA_ACLNAME	string	Yes	
TA_BUFTYPE	string	Yes	
TA_BUFSTYPE	string	Yes	
TA_OBUFTYPE	string	Yes	
TA_OBUFSTYPE	string	Yes	

Configuring the DM_REMOTE_SERVICES Section

The following table lists the fields in the DM_REMOTE_SERVICES section.

Table A-6 DM_REMOTE_SERVICES SECTION

Field Identifier	Field Type	Update	Notes
TA_SERVICENAME	string	No	key
TA_RDOM	string	No	key
TA_LDOM	string	No	key
TA_RNAME	string	Yes	
TA_CONV	string	NoGW	format: { Y | N }
TA_BUFTYPE	string	Yes	
TA_BUFSTYPE	string	Yes	
TA_OBUFTYPE	string	Yes	
TA_OBUFSTYPE	string	Yes	
TA_ROUTINGNAME	string	Yes	
TA_TRANTIME	numeric	Yes	
TA_FUNCTION	string	No	

Configuring the DM_ROUTING Section

The following table lists the fields in the DM_ROUTING section.

Table A-7 DM_ROUTING SECTION

Field Identifier	Field Type	Update	Notes
TA_ROUTINGNAME	string	No	key

Table A-7 DM_ROUTING SECTION

Field Identifier	Field Type	Update	Notes
TA_FIELD	string	Yes	
TA_RANGE	string	Yes	
TA_BUFTYPE	string	Yes	

Configuring the DM_ACCESS_CONTROL Section

The following table lists the fields in the DM_ACCESS_CONTROL section.

Table A-8 DM_ACCESS_CONTROL SECTION

Field Identifier	Field Type	Update	Notes
TA_ACLNAME	string	No	key
TA_RDOM	string	Yes	

Configuring the DM_PASSWORDS Section

The following table lists the fields in the DM_PASSWORDS section.

Table A-9 DM_PASSWORDS SECTION

Field Identifier	Field Type	Update	Notes
TA_LDOM	string	No	key
TA_RDOM	string	No	key
TA_LPWD	string	Yes	format: { Y | N | U }
TA_RPWD	string	Yes	format: { Y | N | U }

The TA_LPWD and TA_RPWD show the existence of a defined password for the local and/or the remote domain. Passwords are not displayed. If an UPDATE operation is selected, the value of the corresponding field must be set to U. The program will then prompt with echo turned off for the corresponding passwords.

Diagnostics in Configuration Mode

dmadmin fails if it cannot allocate an FML typed buffer, if it cannot determine the /etc/passwd entry for the user, or if it cannot reset the environment variables FIELDTBLS or FLDTBLDIR.

The return value printed by dmadmin after each operation completes indicates the status of the requested operation. There are three classes of return values.

The following return values indicate a problem with permissions or a Tuxedo System/T communications error. They indicate that the operation did not complete successfully.

[TAEPERM]
: The calling process specified an ADD, UPDATE, or DELETE operation but it is not running as the System/T administrator. Update operations must be run by the administrator (that is, the user specified in the UID attribute of the RESOURCES section of the TUXCONFIG file).

[TAESYSTEM]
: A Tuxedo System/T error has occurred. The exact nature of the error is written to userlog(3).

[TAEOS]
: An operating system error has occurred.

[TAETIME]
: A blocking timeout occurred. The input buffer is not updated so no information is returned for retrieval operations. The status of update operations can be checked by doing a retrieval on the record that was being updated.

The following return values indicate a problem in doing the operation itself and generally are semantic problems with the application data in the input buffer. The string field TA_STATUS will be set in the output buffer and will contain short text describing the problem. The string field TA_BADFLDNAME will be set to the field name for the field containing the value that caused the problem (assuming the error can be attributed to a single field).

[TAECONFIG]
: An error occurred while reading the BDMCONFIG file.

[TAEDUPLICATE]
: The operation attempted to add a duplicate record.

[TAEINCONSIS]
: A field value or set of field values are inconsistently specified.

[TAENOTFOUND]
: The record specified for the operation was not found.

[TAENOSPACE]
: The operation attempted to do an update but there was not enough space in the BDMCONFIG file.

[TAERANGE]
: A field value is out of range or is invalid.

[TAEREQUIRED]
: A field value is required but not present.

[TAESIZE]
: A field value for a string field is too long.

[TAEUPDATE]
: The operation attempted to do an update that is not allowed.

The following return values indicate that the operation was successful.

[TAOK]
: The operation succeeded. No updates were done to the BDMCONFIG file.

[TAUPDATED]
: The operation succeeded. Updates were made to the BDMCONFIG file.

When using dmunloadcf to print entries in the configuration, optional field values are not printed if they are not set (for strings) or 0 (for integers). These fields will always appear in the output buffer when using dmadmin. In this way, it makes it easier for the administrator to retrieve an entry and update a field that previously was not set. The entry will have the field name followed by a tab but no field value.

Configuration Example

In the following example, `dmadmin` is used to add a new remote domain. For illustration purposes, `ed` is used for the editor.

```
$ EDITOR=ed dmadmin
> config
Sections:
       1) LOCAL_DOMAINS       2) REMOTE_DOMAINS
       3) LOCAL_SERVICES      4) REMOTE_SERVICES
       5) ROUTING             6) ACCESS_CONTROL
       7) PASSWORDS           8) TDOMAIN
       9) OSITP              10) SNA
       11) QUIT
Enter Section [1]: 2
Operations:
       1) FIRST               2) NEXT
       3) RETRIEVE            4) ADD
       5) UPDATE              6) DELETE
       7) NEW_SECTION         8) QUIT
Enter Operation [1]: 4
Enter editor to add/modify fields [n]? y
a
TA_RDOM                       B05
TA_DOMAINID                   BA.BANK05
TA_TYPE                       TDOMAIN
w
53
q
Perform operation [y]? <return>
Return value TAUPDATED
Buffer contents:
TA_OPERATION                  4
TA_SECTION                    2
TA_DOMAINID                   BA.BANK05
TA_RDOM                       B05
TA_TYPE                       TDOMAIN
TA_STATUS                     Update completed successfully
Operations:
       1) FIRST               2) NEXT
       3) RETRIEVE            4) ADD
       5) UPDATE              6) DELETE
       7) NEW_SECTION         8) QUIT
Enter Operation [4]: 7
Sections:
       1) LOCAL_DOMAINS       2) REMOTE_DOMAINS
       3) LOCAL_SERVICES      4) REMOTE_SERVICES
       5) ROUTING              6) ACCESS_CONTROL
       7) PASSWORDS            8) TDOMAIN
       9) OSITP               10) QUIT
Enter Section [1]: 8
Operations:
       1) FIRST                2) NEXT
       3) RETRIEVE             4) ADD
       5) UPDATE               6) DELETE
       7) NEW_SECTION          8) QUIT
Enter Operation [6]: 4
Enter editor to add/modify fields [n]? y
a
TA_RDOM                        B05
TA_NWADDR                      0x00020401c0066d05
w
55
q
Perform operation [y]? <return>
Return value TAUPDATED
Buffer contents:
TA_OPERATION                   4
TA_SECTION                     8
TA_RDOM                        B05
TA_NWADDR                      0x00020401c0066d05
TA_STATUS                      Update completed successfully
Operations:
       1) FIRST                2) NEXT
       3) RETRIEVE             4) ADD
       5) UPDATE               6) DELETE
       7) NEW_SECTION          8) QUIT
Enter Operation [4]: 8
> quit
The dmadmin program ends.
```

Security

If `dmadmin` is run with the application administrator's `UID`, it assumes a trusted user and Security is bypassed. If `dmadmin` is run with another user ID, and if the security option is enabled in the TUXCONFIG file, then the corresponding application password is required to start the `dmadmin` program. If standard input is a terminal, then `dmadmin` will prompt the user for the password with echo turned off. If standard input is not a terminal, the password is retrieved from the environment variable, `APP_PW`. If this environment variable is not specified and an application password is required, then `dmadmin` will fail to start.

When running with another user ID (other than the UID of the administrator) only a limited set of commands is available.

Environment Variables

The `dmadmin` command resets the `FIELDTBLS` and `FLDTBLDIR` environment variables to pick up the ${TUXDIR}/udataobj/dmadmin field table. Hence, the `TUXDIR` environment variable should be set correctly.

If the application requires security and the standard input to `dmadmin` is not from a terminal, then the `APP_PW` environment variable must be set to the corresponding application password.

The `TUXCONFIG` environment variable should be set to the pathname of the Tuxedo System/T configuration file.

General Diagnostics

If the `dmadmin` command is entered before the system has been booted, the following message is displayed:

```
No bulletin board exists. Only logging commands are available.
```

`dmadmin` then prompts for the corresponding commands.

If an incorrect application password is entered or is not available to a shell script through the environment, then a log message is generated, the following message is displayed, and the command terminates:

```
Invalid password entered.
```

Interoperability

`dmadmin` for /SNA must be installed on Tuxedo System/T R6.5. Other nodes in the same domain with an R6.5 gateway may be Tuxedo System/T R4.2.2 or later.

Portability

This command interpreter is supported as a Tuxedo System/T-supplied administrative tool on UNIX and Windows NT operating systems.

See Also

`dmloadcf`(1), `tmadmin`(1), `dmconfig`(5), `DMADM`(5), `addusr`(5), `delusr`(5)

Tuxedo /Domain User Guide

dmconfig

Tuxedo System/T ASCII domain configuration file.

Description

dmconfig is the ASCII version of a Tuxedo System/Domain domain configuration file; it is also referred to by its environmental variable name: DMCONFIG. The dmconfig file is parsed and loaded into a binary version by the dmloadcf utility. The binary configuration file, called the BDMCONFIG file, contains information used by domain gateways to initialize the context required for communications with other domains. dmadmin uses the binary file (or a copy of it) in its monitoring activity. There will be one BDMCONFIG file for each Tuxedo System/Domain application that uses the /Domain feature.

A DMCONFIG file, and its binary BDMCONFIG counterpart, are analogous to the UBBCONFIG and TUXCONFIG files of a non-/Domain System/T application. The DMCONFIG file extends the definition of a non-/Domain System/T application so that the application becomes a domain.

Definitions

A Tuxedo System/Domain Application is defined as the environment described in a single TUXCONFIG file. A System/T Application can communicate with another System/T Application or with another TP Application via a domain gateway group. In “Tuxedo System/Domain” terms, an Application is the same as a TP Domain.

A Gateway Group is a collection of domain gateway processes that provide communication services with a specific type of TP Domain.

A Domain Gateway is a Tuxedo System/Domain process that relays requests and replies to another TP Domain.

A Local Domain characterizes a part of the application (set or subset of services) that is made available to other domains. A Local Domain is always represented by a Domain Gateway Group, and both terms are used as synonyms.

A `Remote Domain` is a remote application that is accessed through a Gateway Group. The remote application may be another Tuxedo System/Domain application or an application running under another TP system.

A `Remote Service` is a service provided by a remote domain that is made available to the local application through a Gateway Group.

A `Local Service` is a service of a local domain that is made available to remote domains through a Gateway Group.

Configuration File Format

The format of a domain configuration file is as follows:

- The file is made up of eight possible specification sections. Lines beginning with an asterisk (*) indicate the beginning of a specification section. Each such line contains the name of the section immediately following the *. Allowable section names are: `DM_LOCAL_DOMAINS`, `DM_REMOTE_DOMAINS`, `DM_SNACRM`, `DM_SNASTACKS`, `DM_SNALINKS`, `DM_LOCAL_SERVICES`, `DM_REMOTE_SERVICES`, `DM_ROUTING`, `DM_ACCESS_CONTROL`, `DM_OSITP`, and `DM_TDOMAIN`. The `DM_LOCAL_DOMAINS` section must precede the `DM_REMOTE_DOMAINS` /.
- Parameters are generally specified by: *`KEYWORD = value`*. This sets *`KEYWORD`* to *`value`*. Valid keywords are described below within each section. *`KEYWORD`*s are reserved; they can not be used as *`value`*s unless they are quoted.

  Lines beginning with the reserved word, `DEFAULT:`, contain parameter specifications that apply to any lines that follow them in the section in which they appear. Default specifications can be used in all sections. They can appear more than once in the same section. The format for these lines is:

  ```
  DEFAULT: [ KEYWORD1 = value1 [ KEYWORD2 = value2 [...]]]
  ```

  The values set on this line remain in effect until reset by another `DEFAULT:` line, or until the end of the section is reached. These values can also be overridden on non-`DEFAULT:` lines by placing the optional parameter setting on the line. If on a non-`DEFAULT:` line, the parameter setting is valid for that line only; lines that follow revert to the default setting. If `DEFAULT:` appears on a line by itself, all previously set defaults are cleared and their values revert to the system defaults.

If a value is `numeric`, standard C notation is used to denote the base (that is, 0x prefix for base 16 (hexadecimal), 0 prefix for base 8 (octal), and no prefix for base 10 (decimal)). The range of values acceptable for a numeric parameter are given under the description of that parameter.

If a value is an `identifier`, standard C rules are used. An `identifier` must start with an alphabetic character or underscore and contain only alphanumeric characters or underscores. The maximum allowable length of an identifier is 30 (not including the terminating null). An identifier cannot be the same as any `KEYWORD`.

A value that is neither an integer number or an identifier must be enclosed in double quotes. Certain special characters can be escaped inside a string using a backslash. “\\’’ translates to a single backslash. ‘‘\“\“’’ translates to a double quote. “\n” translates to a newline. “\t” translates to a tab. “\f” translates to a form feed. “\x” (where ’x’ is any character other than one of the previously mentioned special characters) translates to ’x’.

- Input fields are separated by at least one space (or tab) character.
- “#” introduces a comment. A newline ends a comment.
- Blank lines and comments are ignored.
- Comments can be freely attached to the end of any line.
- Lines are continued by placing at least one tab after the newline. Comments can not be continued.

VERSION=`string_value`

where `string_value` can be any value. The field is not checked by the software; it is provided simply as a place where the customer can enter a string that may have some documentation value to the application.

The DM_LOCAL_DOMAINS Section

This section identifies local domains and their associated gateway groups. The section must have an entry for each gateway group (Local Domain). Each entry specifies the parameters required for the domain gateway processes running in that group.

Entries have the form:

`LDOM` required parameters [optional parameters]

where `LDOM` is an *`identifier`* value used to name each local domain. `LDOM` must be unique within a particular configuration. As you will see in the description of the DM_LOCAL_SERVICES section, `LDOM` is the identifier that connects local services with a particular gateway group.

The following are the required parameters:

`GWGRP = ` *`identifier`*

Specifies the name of the gateway server group (the name provided in the `TUXCONFIG` file) representing this local domain. There is a one-to-one relationship between a *`DOMAINID`* (see below) and the name of the gateway server group, that is, each `GWGRP` must have its own, unique `DOMAINID`.

`TYPE = ` *`identifier`*

Is used for grouping local domain into classes. `TYPE` can be set to one of the following values: `TDOMAIN`, `OSITP` or `SNAX`. The `TDOMAIN` value indicates that this local domain can only communicate with another Tuxedo System/Domain. The `OSITP` value indicates that this local domain communicates with another TP Domain via the OSI-TP protocol. The `SNA` value indicates that this local domain communicates with an MVS/CICS region via the LU6.2 protocol. Domain types must be defined in the `$TUXDIR/udataobj/DMTYPE` file.

`DOMAINID = ` *`string`*

Is used to identify the local domain. `DOMAINID` must be unique across both local and remote domains. The value of *`string`* can be a sequence of characters (for example, "BA.CENTRAL01"), or a sequence of hexadecimal digits preceded by “0x” (for example, “0x0002FF98C0000B9D6”). `DOMAINID` must be 32 octets or fewer in length. If the value is a string, it must be 32 characters or fewer (counting the trailing null).

`DMTLOGDEV = ` *`string`*

Specifies the Tuxedo file system that contains the Domain transaction log (`DMTLOG`) for this machine. The `DMTLOG` is stored as a Tuxedo System VTOC table on the device. If this parameter is not specified (and it should not be specified if `TYPE=SNADOM`), the domain gateway group is not allowed to process requests in transaction mode. Local domains running on the same machine can share the same `DMTLOGDEV` file system, but each local domain must have its own log (a table in the `DMTLOGDEV`) named as specified by the `DMTLOGNAME` keyword (see below).

Optional parameters describe resources and limits used in the operation of domain gateways:

`AUDITLOG = `*`string`*
: Specifies the name of the audit log file for this local domain. The audit log feature is activated from the `dmadmin`(1) command and records all the operations within this local domain. If the audit log feature is active and this parameter is not specified, the file DM`mmddyy`.LOG (where `mm`=month, `dd`=day, and `yy`=year) is created in the directory specified by the `$APPDIR` environment variable or the `APPDIR` keyword of the MACHINES section of the `TUXCONFIG` file.

`BLOCKTIME = `*`numeric`*
: Specifies the maximum wait time allowed for a blocking call. The value sets a multiplier of the `SCANUNIT` parameters specified in the `TUXCONFIG` file. The value `SCANUNIT * BLOCKTIME` must be greater than or equal to `SCANUNIT` and less than 32,768 seconds. If this parameter is not specified, the default value is set to the value of the `BLOCKTIME` parameter specified in the `TUXCONFIG` file. A time-out always implies a failure of the affected request. Notice that the time-out specified for transactions in the `TUXCONFIG` will always be used when the request is issued within a transaction.

`DMTLOGNAME = `*`identifier`*
: Specifies the name of the domain transaction log for this domain. This name must be unique when the same `DMTLOGDEV` is used for several local domains. If not specified, the default is the string ‘‘`DMTLOG`’’. The name must be 30 characters or less. Since transactions are not support for SNA-type gateways, this parameter has no meaning when `TYPE=SNADOM`.

`DMTLOGSIZE = `*`numeric`*
: Specifies the numeric size, in pages, of the Domain transaction log for this machine. It must be greater than 0 and less than the amount of available space on the Tuxedo file system. If not specified, the default is 100 pages. Since transactions are not support for SNA-type gateways, this parameter has no meaning when `TYPE=SNADOM`.

`MAXDATALEN = `*`numeric`*
: Specifies a maximum amount of data (in bytes) that can be sent to or from any services advertised by this local domain. There is no limit if this parameter is not specified.

`MAXRDOM =` *`numeric`*
: Specifies the maximum number of connections (or dialogues if the domain is of type *`OSITP`*) allowed per gateway. There is no limit if this parameter is not specified.

`MAXRDTRAN =` *`numeric`*
: Specifies the maximum number of domains that can be involved in a transaction. It must be greater than 0 and less than 32,768. If not specified, the default is 16. Since transactions are not support for SNA-type gateways, this parameter has no meaning when `TYPE=SNADOM`.

`MAXTRAN =` *`numeric`*
: Specifies the maximum number of simultaneous global transactions allowed on this local domain. It must be greater than or equal to 0 and less than or equal to the `MAXGTT` parameter specified in the `TUXCONFIG` file. If not specified, the default is the value of `MAXGTT`.

`MAXSENDLEN =` *`numeric`*
: Specifies the maximum length (in bytes) of messages sent or received by this local domain. If this parameter is set all messages sent or received will be broken up into packets of no more than MAXSENDLEN bytes. There is no limit if this parameter is not specified.

`SECURITY =` *`value`*
: Specifies the type of application security to be enforced. The following description applies to security in an SNA-type gateways.

The combined settings of the `SECURITY` parameters in the `UBBCONFIG` and the `DMCONFIG` files have the following effects:

- When the `DM_LOCAL_DOMAINS` Security parameter is set to `NONE` or `APP_PW`, no action is taken by the eAM gateway with regard to security.
- However, when the `UBBCONFIG` file Security parameter is set to `APP_PW`, the application password is validated by an `AUTHSVC` when clients join the application. The `AUTHSVC` is provided by the user application.

If security is to be enforced by both the local domain and the host system for each request outbound from the local domain, the following settings must be made:

- `UBBCONFIG` file Security parameter must be set to one of: `USER_AUTH`, `ACL`, or `MANDATORY_ACL`
- `DMCONFIG` file `DM_LOCAL_DOMAINS` section Security parameter must be set to `DM_USER_PW`

- DM_SNALINKS Security parameter must be set to IDENTIFY or VERIFY.

If security is to be enforced by both the local domain and the host system for each request inbound from the host system to the local domain, the following settings must be made:

- UBBCONFIG file Security parameter must be set to one o: USER_AUTH, ACL, or MANDATORY_ACL
- DMCONFIG file DM_LOCAL_DOMAINS section Security parameter must be set to DM_USER_PW
- DM_SNALINKS Security parameter must be set to IDENTIFY or VERIFY.

For a request sent to the host system, the local principal userid is located in the domain security table and the associated remote userid, or userid and password, are put into the conversation start-up request before being sent over the LU6.2 conversation. (This occurs if SECURITY is set to IDENTIFY or VERIFY in the DM_SNALINKS section of the DMCONFIG file.)

For requests sent from the host system, the local domain extracts the remote userid, or userid and password, from the conversation start-up request and checks the domain security table. That table contains pairs of local principal user IDs and remote user IDs, maintained on a service-by-service basis. The remote user ID is mapped to the local principal userid. The local principal userid and password are used for further Access Control List (ACL) checking, as specified in the UBBCONFIG file.

When a request is received from the host system, the local domain checks the DMCONFIG file ACL for the local service to see if requests from the remote domain are permitted. If the DMCONFIG file does not contain an ACL for the local service, the service is accessible to all requests.

Therefore, if the ATTACHSEC level for the connection definition in the host system is Identify or Verify, the DMCONFIG SECURITY parameter must be set to DM_USER_PW so that a userid and a password are sent on the conversation start-up requests.

The DM_REMOTE_DOMAINS Section

This section identifies the known set of remote domains and their characteristics.

Entries have the form:

`RDOM` `required parameters` [optional parameters]

where *`RDOM`* is an *`identifier`* value used to identify each remote domain known to this configuration. *`RDOM`* must be unique within the configuration.

The following parameters are required:

`TYPE =` *`identifier`*

Is used for grouping remote domain into classes. `TYPE` can be set to one of the following values: `TDOMAIN`, `OSITP` or `SNAX`. The `TDOMAIN` value indicates that this remote domain can only communicate with another Tuxedo System/Domain. The `OSITP` value indicates that this remote domain communicates with another TP domain via the OSI-TP protocol. The `SNAX` value indicates that this remote domain communicates with an MVS/CICS region via the LU6.2 protocol.

`DOMAINID =` *`string`*

Is used to identify a remote domain. `DOMAINID` must be 32 octets or fewer in length. If the value is a string, it must be 32 characters or fewer (counting the trailing null). `DOMAINID` must be unique across remote domains. The value of *`string`* can be a sequence of characters or a sequence of hexadecimal digits preceded by “`0x`”.

The following parameter is optional:

`CODEPAGE =` *`“table identifier”`*

Is used to designate a bidirectional translation table for ASCII to EBCDIC conversion between a local Tuxedo application and a remote mainframe application. The table identifier describes a file containing a translation table and must be enclosed by double quotes. The name of the file, located in the `$TUXDIR/udatajobj/codepage` directory, is a composite of the code page numbers used for the translation, for example:
`CODEPAGE=”00819x00297”`
designates the translation table for converting ASCII CP-00819 characters to French EBCDIC CP-00297 characters, and vice versa. The translation tables can be modified. Refer to Appendix F, “Code Page Translation Tables,” for complete character listings.

The DM_TDOMAIN Section

This section defines the addressing information required by domains of type `TDOMAIN`. This section should have an entry per local domain if requests from remote domains to local services are accepted on that local domain (gateway group), and an entry per remote domain accessible by the defined local domains.

Entries have the form:

```
DOM   required parameters [optional parameters]
```

where *`DOM`* is an *`identifier`* value used to identify either a local domain (`LDOM`) or a remote domain (`RDOM`) in the DM_LOCAL_DOMAINS section or in the DM_REMOTE_DOMAINS section. The *`DOM`* identifier must match a previously defined *`LDOM`* in the DM_LOCAL_DOMAINS sections or *`RDOM`* in the DM_REMOTE_DOMAINS section.

The following parameter is required:

`NWADDR =` *`string`*

This parameter specifies the network address used by a local or a remote domain to accept connections from other Tuxedo System/Domain Domains. If *`string`* has the form *`''0xhex-digits''`*, it must contain an even number of valid hexadecimal digits.

The following parameter is optional:

`NWIDLETIME = numeric`

This parameter specifies the maximum time allowed for a connection to be idle (that is, unused). When this time is reached, the idle connection is be terminated. The numeric value represents a time in minutes. If this keyword is not specified, then idle connections will be maintained until the gateway handling the connection is shutdown.

Notice that multiple entries for a particular domain may be defined in this table. Multiple addresses specified for a remote domain mean that the first address (the first entry in the table for the remote domain) should be used to establish the connection and the other addresses should be used as back-up addresses in case of failure of the connection setup to the first address. Multiple addresses specified for a local domain mean that multiple listening ports are available on the same or different types of networks.

The DM_OSITP Section

This section defines the addressing information required by domains of type `OSITP`. This section should have one entry per gateway group (local domain), and one entry per remote domain of type `OSITP`.

Entries have the form:

```
DOM   required parameters [optional parameters]
```

where *`DOM`* is an *`identifier`* value used to identify a local domain (LDOM) or a remote domain (RDOM) in the DM_LOCAL_DOMAINS section or in the DM_REMOTE_DOMAINS section. The *`DOM`* identifier must match a previously defined *`LDOM`* in the DM_LOCAL_DOMAINS sections or *`RDOM`* in the DM_REMOTE_DOMAINS section.

The following are required parameters:

`APT =` *`string`*
: This parameter specifies an OSI Application Process Title (APT). An APT may be a name (i.e., the Directory Name of an Application Process Title) or an object identifier (i.e., a sequence of integer values separated by periods).

`AEQ =` *`string`*
: This parameter specifies an OSI Application Entity Qualifier (AEQ). An AEQ may be a name (i.e., the relative distinguished name of a particular Application Entity) or an integer (i.e., if the APT is an object identifier).

The following are optional parameters:

`AET =` *`string`*
: This parameter specifies an OSI Application Entity Title (AET). An AET is formed from an Application Process Title (APT) and an Application Entity Qualifier (AEQ), i.e. in ASN.1 AET is defined as a SEQUENCE { APT, AEQ } where APT and AET are of type ANY. Three main formats are accepted for the value of *`string`*:

  `encoded string`
  : This is a single value as a hexadecimal octet string which a represents a valid BER encoding of the AET, e.g. AET = “0x06062B80CE0F0107”.

`{object identifier}, {integer}`

The first element represents the APT defined as an object identifier (i.e., a sequence of integer values separated by periods) and the second element represents an AEQ defined as an integer constant, e.g., AET = "{1.3.15.0.3},{1}".

`{string}, {string}`

This format allows the APT and the AEQ to be defined as string constants, e.g., AET = "{BA.CENTRAL01},{Tuxedo}".

`ACN =` *`{XATMI | UDT}`*

This parameter specifies the object identifier of the Application Context Name (ACN) used by this domain. Current allowed application contexts are: the `XATMI-ASE` (XATMI) and the `UDT-ASE` (UDT). If this parameter is not specified, the ACN is set to the object identifier of the `XATMI-ASE` Application Context.

`APID =` *`integer`*

This parameter specifies an OSI Application Process Invocation Identifier (APID).

`AEID =` *`integer`*

This parameter specifies an OSI Application Entity Invocation Identifier (AEID).

`PROFILE =` *`identifier`*

This parameter specifies the OSI TP profile used by this domain and is used to determine the required OSI TP functional units. `PROFILE` can be set to one of the following values: `ATP11`, `ATP21`, `ATP31`, `ATP12`, `ATP22`, and `ATP32`. The UDT ASE application context allows the use of any of these profiles. The XATMI-ASE application context only allows profiles `ATP11`, `ATP21` and `ATP31`. Profiles `ATP11`, `ATP21` and `ATP31` use the Dialogue, Polarized Control and Handshake functional units. Profiles `ATP12`, `ATP22` and `ATP32` use the Dialogue, Shared Control, and Handshake functional units. Profiles `ATP11` and `ATP12` do not use OSI TP transactions (the Commit functional unit is not used). Profiles `ATP21` and `ATP22` require the Commit, Unchained Transactions, and Recovery functional units. Profiles `ATP31` and `ATP32` require the Commit, Chained Transactions, and Recovery functional units. By default, the `ATP21` profile is always selected.

`URCH =` *`string`*

This parameter specifies the user portion of the OSITP Recovery Context Handle. It may be required by the XAP-TP provider in order to perform recovery of distributed transactions after a communications line or system failure.

The DM_SNACRM Section

The DM_SNACRM section provides three (3) keywords used to identify the Communications Resource Manager that will provide ATMI transaction semantics between a given domain and it's partners. Entries have the general form:

<CommunicationsResourceManagerName> parameters

Where <CommunicationsResourceManagerName> is the locally known name of this `SNACRM` definition to be used when referencing this `SNACRM` in subsequent sections. This name is an ASCII string 1 to 30 characters in length. The parameters are the keyword/value pairs that makeup the definition. All keywords are required for a valid `SNACRM` definition. Keywords can be in any order.

LDOM <LocalDomainName>

LDOM associates this `SNACRM` with a defined local domain. <LocalDomainName> is the reference to an entry in the DM_LOCAL_DOMAINS section. This name is an ASCII string 1 to 30 characters in length. This parameter is required. This parameter has no default.

SNACRMADDR <HexSocketAddress> or <//host:port>

SNACRMADDR provides the socket address the eAM gateway uses to communicate with the `SNACRM`. If the `SNACRM` is started independent of the gateway, this address must be used on the `SNACRM` command line. <HexSocketAddress> is a TCP/IP address using the sockaddr_in format of family,port,address:

`<0xFFFFPPPPAAAAAAAA>`

where:

`FFFF`

is the hex value of the protocol family, always 0x0002 for the INET family.

`PPPP`

is the hex value of an unused TCP/IP port

`AAAAAAAA`
: is the hex value of the IP address for the machine running the `SNACRM`

Therefore if the `SNACRM` was running on a machine with an IP address of 206.189.43.13, and we wanted to use port 6000 for the `SNACRM` then SNACMADDR would be:

```
0x00021770CEBD2B0D
```

This parameter is required. This parameter must contain an even number of hex characters. This parameter has no default.

The DM_SNASTACKS Section

The DM_SNASTACKS section provides five (5) keywords which identify the third party SNA stack that should be used for connections established between a given domain and it’s partners. Entries have the general form:

<StackReference>   parameters

Where <StackReference> is the locally known name of this stack definition to be used when referencing this stack in subsequent sections. This name is an ASCII string 1 to 30 characters in length. The parameters are the keyword/value pairs that makeup the definition. All keywords are required for a valid stack definition. Keywords can be in any order.

LOCALLU <LocalLUAlias>
: LOCALLU provides a reference to an LU alias defined in the third party SNA stack. <LocalLUAlias> is the name used to identify the local LU definition as specified by the third party SNA stack configuration. This is a name that represents the end node for an LU6.2 connection. The value for this parameter is an ASCII string, 1 to 64 characters in length. This parameter is required. This parameter has no default. The third party SNA stack will require a corresponding definition for a local LU.

LTPNAME <LocalTransactionProgramName>
: LTPNAME identifies the inbound transaction programs which will be serviced by any CRM using this stack definition. <LocalTransactionProgramName> is the name used to identify inbound transaction programs for which an attach will be accepted. The only useful

value is an asterisk. This indicates all inbound attaches will be accepted. This parameter is required. This parameter has no default. Partial TP names are not supported. The third party SNA stack will require a corresponding definition for inbound TP names.

SNACRM <CommunicationsResourceMangerName>
: SNACRM provides a name by which to reference the associated CRM definition. <CommunicationsResourceMangerName> is the name used to associate the DM_SNACRM definition with this DM_SNASTACKS entry. The value for this parameter is an ASCII string, 1 to 32 characters in length. This parameter is required. This parameter has no default.

STACKPARMS <parameters required for third party sna stack>
: STACKPARMS provides a method for the domain gateway to pass any required parameters to the third party SNA stack. <parameters required for third party sna stack> is an ASCII string, 1 to 128 characters in length. Currently, the only value used is the TCP/IP hostname for the machine running the third party SNA stack. This parameter is required. This parameter has no default.

STACKTYPE={ hp60 | ibm60 | spx62 | sun91 | ms40 | vtam28}
: This option is used to indicate which vendor SNA stack is being used. It is also used to determine the name of specific BEA eLink Adapter for Mainframe system libraries. It is essential that the value of this option be coded correctly. These values are mapped to the equivalent BEA eLink Adapter for Mainframe system library.

The DM_SNALINKS Section

This section defines the SNA Link information required by domains of type SNA. Entries have the form:

LINK parameters

Where *LINK* is an *identifier* value used to identify a connection between a local domain (LDOM) and a remote domain (RDOM). The *RDOM* identifier must match a previously defined *RDOM* in the DM_REMOTE_DOMAINS section.

The following parameters are available:

STACKREF = *string*

This required parameter defines the stack that will be used for establishment of this link. The STACKREF string is the tag that was used in a previous definition established in the DM_SNASTACKS section.

RDOM = *string*

The RDOM string should match a previous RDOM definition in the DM_REMOTE_DOMAINS section.

LSYSID = <Connection ID of remote (CICS) region>

LSYSID is the 4 character identifier that is to be used for this link. This should match the connection ID used by a partner CICS to communicate to the CRM across this link.

RSYSID = <SYSID of remote (CICS) region>

RSYSID is the 4 character remote sysid of the partner. Typically it is the sysid of a CICS region, but could also be the subsystem id of an IMS control region. This parameter should match the actual sysid of the remote partner.

RLUNAME = <Alias of APPLID for remote region>

The RLUNAME value represents an alias known to the third party SNA stack that resolves to a VTAM netname for the remote application. This would most likely be the VTAM applid for a CICS region, however it could also be an APPC/MVS LU defined for use with IMS. The value must be unique within the SNA network. *string* should be from 1 to 8 characters. This parameter is required. This parameter has no default. The third party stack configuration requires a matching definition.

MODENAME = <Mode name VTAM mode entry>

MODENAME is VTAM mode entry, defined to the third party SNA stack, to be used for this link. For a CICS link this must be compatible with the RDO session definition for the corresponding connection. For an IMS connection this must be compatible with the DLOGMOD entry on the LU definition used to access the IMS scheduler. *string* should be from 1 to 8 ASCII characters. This parameter is required. This parameter must match the third party SNA stack configuration and must be compatible with the corresponding entries defined to VTAM and/or CICS.

SECURITY = *string*

SECURITY_TYPE specifies the security setting in CICS/RACF or partner. Legal values are LOCAL, IDENTIFY, VERIFY, PERSISTENT or MIXIDPE. *string* should be from 1 to 10 characters. The default setting is LOCAL.

MAXSESS = number

MAXSESS is the maximum number of parallel sessions that can be started on this link. MAXSESS must be greater than or equal to four.

MINWIN = number

The minimum number of contention winners. This value is typically half the MAXSESS value.

MAXSYNCLVL = number

This value represents the maximum transaction synchronization level that can be supported over this link.

A value of zero (0) means this link is non-transactional. No synchronization will be maintained. This level can be used for sending and receiving messages from IMS via the APPC/MVS transparency interface. The default sync-level is sync-level 0.

A value of one (1) means this link will support everything supported with zero (0), in addition to:

Outbound ATMI tpcall() as a CICS distributed program link request with the semantics of SYNCONRETURN.
Inbound EXEC CICS LINK requests with the semantics of SYNCONRETURN. The program name must match the RNAME on the local service definition and the SYSID must match the LSYSID for the link.

A value of two (2) means this link will support everything supported with zero (0) and one (1) for partners able to exchange logs and compare states, in addition to:

The exchange logs and compare states function with a partner CICS.
Outbound ATMI tpcall() as a CICS distributed program link request with full two phase commit transaction semantics using tpcommit().
Outbound ATMI tpconnect() as APPC or CPIC distributed transaction processing with full two phase commit transaction semantics using tpcommit().
Inbound EXEC CICS LINK requests with full two phase commit transaction semantics using Prepare Rollback and Syncpoint verbs.
Inbound APPC or CPIC conversations with full two phase commit transaction semantics using Prepare Rollback and Syncpoint verbs.

The partner must be able to negotiate a CICS style exchange logs and compare states for successful initialization of a sync-level 2 link.

If the installation is not licensed for sync-level 2, this parameter must be set to 0 or 1 for the link to be established. Transaction support is only available at sync-level 2. Distributed Program Link can be accessed as SYNCONRETURN, that is, not transactional if the link sync-level is 1.

Caution: If you set MAXSYNCLVL=2 or make no entry for this parameter (that is, accept the default) without having installed the eAM software licensed for that level, the system configuration automatically reverts to Sync-level 1 and an error message is sent to the error log. To clear that error message, you must either reset the MAXSYNCLVL parameter to an appropriate value or purchase and install the correct software.

STARTTYPE = {auto | cold}

This option sets the recovery mode for transactional links. When set to AUTO, the system restarts using configuration and link data recovered from the in-flight transaction log. When set to COLD, the system uses configuration data taken from the current dmconfig file and loses any in-flight link data. Changing dmconfig file parameters and performing an AUTO start results in a message warning that changed parameters are ignored until the next cold start. To force a cold start and disregard the STARTTYPE setting, delete the SNA*LOG files in $APPDIR.

The DM_ACCESS_CONTROL Section

This section specifies the access control lists used by local domain. Lines in this section are of the form:

`ACL_NAME` `required parameters`

where *`ACL_NAME`* is a (*`identifier`*) name used to identify a particular access control list; it must be 15 characters or less in length.

Required parameters are:

`ACLIST =` *`identifier`* `[,`*`identifier`*`]`

where an `ACLIST` is composed of one or more remote domain names (`RDOM`) separated by commas. The wildcard character (*) can be used to specify that all the remote domains defined in the DM_REMOTE_DOMAINS section can access a local domain.

The DM_LOCAL_SERVICES Section

This section provides information on the services exported by each local domain. This section is optional and if it is not specified then all local domains defined in the DM_LOCAL_DOMAINS section accept requests to all of the services advertised by the Tuxedo System/Domain application. If this section is defined then it should be used to restrict the set of local services that can be requested from a remote domain.

Lines within this section have the form:

```
service    [optional parameters]
```

where `service` is the (`identifier`) local name of the exported service, and it must be 1-15 characters in length. This name corresponds to a name advertised by one or more servers running with the local Tuxedo System/Domain application. Notice that exported services inherit the default or special properties specified for the service in an entry in the `SERVICES` section of the `TUXCONFIG` file. Some of these parameters are: `LOAD`, `PRIO`, `AUTOTRAN`, `ROUTING`, `BUFTYPE`, and `TRANTIME`.

Optional parameters are:

`ACL = ` *`identifier`*
: Specifies the name of the access control list (ACL) to be used by the local domain to restrict requests made to this service by remote domains. The name of the ACL is defined in the DM_ACCESS_CONTROL section. If this parameter is not specified then access control will not be performed for requests to this service.

`API = ATMI`
: Specifies the API used by the local service. Currently the only supported value is ATMI. This parameter is required.

`CONV = { Y | N }`
: Specifies whether (`Y`) or not (`N`) the local service is a conversational service. The default value is N.

`LDOM = ` *`identifier`*
: Specifies the name identifying the local domain exporting this service. If this keyword is not specified, then the first local domain entry in the DM_LOCAL_DOMAINS section accepts requests for this local service.

`INBUFTYPE = ` *`type[:subtype]`*
: Restricts the buffer type naming space of data types accepted by this service to a single buffer type. This parameter should be defined when the service is

going to be used from an `OSITP` type gateway that uses the UDT ASE Application Context. For SNA-type gateways buffer types, see the discussion in the DM_REMOTE_SERVICES section below.

`OUTBUFTYPE =` *`type[:subtype]`*

Restricts the buffer type naming space of data types returned by this service to a single buffer type. This parameter should be defined when the service is going to be used from an `OSITP` type gateway that uses the UDT ASE Application Context. The `FML` buffer type cannot be used for `OSITP` type gateways. For SNA-type gateways buffer types, see the discussion in the DM_REMOTE_SERVICES section below.

`RNAME =` *`string`*

The `RNAME` option is the local-service name imported from a remote CICS/ESA region. This name is used by the CRM to select a local service. When the `RNAME` specifies an alternate mirror transaction identifier for explicit attachment for inbound DPL requests, it must be a combination of the alternate mirror `TRANSID` and a CICS/ESA program name in the following format:

```
RNAME=AAAA:BBBBBBBB
```

where:

`AAAA`

is a 1-4 character alternate mirror TRANSID

`BBBBBBBB`

is a 1-8 character CICS/ESA program name.

The colon is required to indicate the `TRANSID`/program name combination. The `TRANSID` must be composed of acceptable CICS/ESA characters:

```
A-Za-z0-9$@#./-_%&Q¢?!|”=,;<>
```

The DM_REMOTE_SERVICES Section

This section provides information on services "imported" and available on remote domains. Lines within this DM_REMOTE_SERVICES section have the form:

```
service    [optional parameters]
```

where `service` is the (`identifier`) name used by the local Tuxedo System/Domain application for a particular remote service. Remote services are associated with a particular remote domain.

Optional Parameters are:

`AUTOTRAN = { Y | N }`
: Specifies whether or not a transaction should automatically be started if a request message is received that is not already in transaction mode. The default is N.

`BLOCKTIME = numeric`
: Specifies the maximum wait time allowed for a reply to this remote service. The value sets a multiplier of the SCANUNIT parameters specified in the TUXCONFIG file. The value SCANUNIT * BLOCKTIME must be greater than or equal to SCANUNIT and less than 32,768 seconds. A time-out always implies a failure of the affected transaction or request.

`CONV = { Y | N }`
: Specifies whether (`Y`) or not (`N`) the remote service is a conversational service. The default value is N.

`FUNCTION = {APPC|DPL}`
: Enables outbound Tuxedo service requests to map to APPC transaction programs or CICS programs. The default value `APPC` indicates the remote service is a transaction program that may or may not be running under CICS. The `DPL` value indicates the remote service maps to a program running under CICS.

`LDOM = identifier`
: Specifies the name of a local domain in charge of routing requests to this remote service. The gateway group associated with the local domain advertises `service` in the Tuxedo System/Domain Bulletin Board. If this parameter is not specified then all the local domains will be able to accept requests to this remote service. The service request will be then redirected to a remote domain of the same type (see `RDOM` keyword below).

`LOAD =` *`integer`*
: Specifies that the remote service imposes a load of integer units. The value of `LOAD` can be between 1 and 32767 inclusive. If not specified, the default is 50. A higher number indicates a greater load.

`INBUFTYPE =` *`type[:subtype]`*
: Restricts the buffer type naming space of data types accepted by this service to a single buffer type. This parameter should be defined when the service is going to be used from an `OSITP` type gateway that uses the UDT ASE Application Context. The `FML` buffer type cannot be used for `OSITP` type gateways.

`OUTBUFTYPE =` *`type[:subtype]`*
: Restricts the buffer type naming space of data types returned by this service to a single buffer type. This parameter should be defined when the service is going to be used from an `OSITP` type gateway that uses the UDT ASE Application Context. The `FML` buffer type cannot be used for `OSITP` type gateways.

`PRIO = integer`
: Specifies the dequeing priority of service requests to this remote service. The value of `PRIO` must be greater than 0 and less than or equal to 100, with 100 being the highest priority. The default is 50.

`RDOM =` *`identifier`*
: Specifies the name of the remote domain responsible for the actual execution of this service. If this parameter is not specified and a routing criteria (see below `ROUTING` keyword) is not specified, then the local domain assumes that any remote domain of the same type accepts this service and it selects a known domain (a domain to which a connection already exists) or remote domain from the \DM_REMOTE_DOMAINS section.

`RNAME =` *`string`*
: Specifies the actual service name expected by the remote domain. If this parameter is not specified, the remote service name is the same as the name specified in *`service`*.

  The `RNAME` option is the name of the host TP_NAME. For non-CICS systems, this name can be up to 64 characters in length. For CICS systems, this name is the trans-id name for APPC-defined requests and the program name for DPL requests. CICS trans-id names cannot exceed four characters and CICS program names cannot exceed eight characters. The RNAME option must observe these requirements.

When the `RNAME` specifies an alternate mirror transaction identifier for explicit attachment to outbound DPL requests, it must be a combination of the alternate mirror `TRANSID` and an advertised remote CICS/ESA program name in the following format:

`RNAME=AAA:BBBBBBBB`

where:

`AAA`

is a 1-4 character alternate mirror TRANSID

`BBBBBBBB`

Is a 1-8 character CICS/ESA program name.

The colon is required to indicate the `TRANSID`/program name combination. The `TRANSID` must be composed of acceptable characters recognized in CICS/ESA identifiers:

`A-Za-z0-9$@#./-_%&Q¢?!|”=,;<>`

`ROUTING = `*`identifier`*
When more than one remote domain offers the same service, a local domain can perform data dependent routing if this optional parameter is specified. The *`identifier`* specifies the name of the routing criteria used for this data dependent routing. If not specified, data dependent routing is not done for this service. *`identifier`* must be 15 characters or less in length. If multiple entries exist for the same service name but with different `RDOM` parameters, the `ROUTING` parameter should be the same for all of these entries.

`TRANTIME = `*`integer`*
specifies the default time-out value in seconds for a transaction automatically started for the associated service. The value must be greater than or equal to 0 and less than 2147483648. The default is 30 seconds. A value of 0 implies the maximum time-out value for the machine.

The DM_ROUTING Section

This section provides information for data dependent routing of /T Domain service requests using `FML`, `VIEW`, `X_C_TYPE`, and `X_COMMON` typed buffers. Lines within the `DM_ROUTING` section have the form:

```
CRITERION_NAME    required parameters
```

where *CRITERION_NAME* is the (*identifier*) name of the routing entry that was specified on the services entry. *CRITERION_NAME* must be 15 characters or less in length.

Required parameters are:

`FIELD =` *identifier*
: Specifies the name of the routing field. It must be 30 characters or less. This field is assumed to be a field name that is identified in an `FML` field table (for `FML` buffers) or an `FML` view table (for `VIEW`, `X_C_TYPE`, or `X_COMMON` buffers). The `FLDTBLDIR` and `FIELDTBLS` environment variables are used to locate `FML` field tables, and the `VIEWDIR` and `VIEWFILES` environment variables are used to locate `FML` view tables.

`RANGES =` *string*
: Specifies the ranges and associated remote domain names (`RDOM`) for the routing field. *string* must be enclosed in double quotes. The format of *string* is a comma-separated ordered list of range/RDOM pairs (see `EXAMPLES` below).

A range is either a single value (signed numeric value or character string in single quotes), or a range of the form “lower - upper” (where lower and upper are both signed numeric values or character strings in single quotes). Note that “lower” must be less than or equal to “upper”. To embed a single quote in a character string value (as in `O'Brien`, for example), it must be preceded by two backslashes (`'O'Brien'`). The value MIN can be used to indicate the minimum value for the data type of the associated `FIELD`; for strings and arrays, it is the null string; for character fields, it is 0; for numeric values, it is the minimum numeric value that can be stored in the field. The value `MAX` can be used to indicate the maximum value for the data type of the associated `FIELD`; for strings and arrays, it is effectively an unlimited string of octal-255 characters; for a character field, it is a single octal-255 character; for numeric values, it is the maximum numeric value that can be stored in the field. Thus, “`MIN - -5"` is all numbers less than or equal to –5 and “`6 - MAX"` is all numbers greater than or equal to 6. The meta-character “*’’ (wild-card) in the position of a range indicates any values not covered by the other ranges previously seen in the entry; only one wild-card range is allowed per entry and it should be last (ranges following it will be ignored).

The routing field can be of any data type supported in `FML`. A numeric routing field must have numeric range values and a string routing field must have string range values.

String range values for string, array, and character field types must be placed inside a pair of single quotes and can not be preceded by a sign. Short and long integer values are a string of digits, optionally preceded by a plus or minus sign. Floating point numbers are of the form accepted by the C compiler or `atof():` an optional sign, then a string of digits optionally containing a decimal point, then an optional e or E followed by an optional sign or space, followed by an integer.

When a field value matches a range, the associated `RDOM` value specifies the remote domain to which the request should be routed. A `RDOM` value of "*" indicates that the request can go to any remote domain known by the gateway group.

Within a range/`RDOM` pair, the range is separated from the `RDOM` by a ":".

`BUFTYPE = ~`*`type1`*`[`*`:subtype1`*`[,`*`subtype2`*` . . . ]][;`*`type2`*`[`*`:subtype3`*`[, . . . ]]] . . .~`

Is a list of types and subtypes of data buffers for which this routing entry is valid. The types are restricted to be either `FML`, `VIEW`, `X_C_TYPE`, or `X_COMMON`. No subtype can be specified for type `FML` and subtypes are required for the other types ("*" is not allowed). Duplicate type/subtype pairs can not be specified for the same routing criterion name; more than one routing entry can have the same criterion name as long as the type/subtype pairs are unique. This parameter is required. If multiple buffer types are specified for a single routing entry, the data types of the routing field for each buffer type must be the same.

If the field value is not set (for `FML` buffers), or does not match any specific range and a wild-card range has not been specified, an error is returned to the application process that requested the execution of the remote service.

Files

The `BDMCONFIG` environment variable is used to find the BDMCONFIG configuration file.

Example 1

The following configuration file defines a 5-site domain configuration. The example shows 4 Bank Branch domains communicating with a Central Bank Branch. Three of the Bank Branches run within other Tuxedo System/Domain domains. The fourth Branch runs under the control of another TP Domain and OSI-TP is used in the communication with that domain.

```
# Tuxedo DOMAIN CONFIGURATION FILE FOR THE CENTRAL BANK
#
#
*DM_LOCAL_DOMAINS
# <local domain name> <Gateway Group name> <domain type> <domain
id> <log device>
#                      [<audit log>] [<blocktime>]
#                      [<log name>] [<log offset>] [<log size>]
#                      [<maxrdom>] [<maxrdtran>] [<maxtran>]
#                      [<maxdatalen>] [<security>]
#                      [<tuxconfig>] [<tuxoffset>]
#
#
DEFAULT: SECURITY = NONE

c01    GWGRP = bankg1
       TYPE = TDOMAIN
       DOMAINID = "BA.CENTRAL01"
       DMTLOGDEV = “/usr/apps/bank/DMTLOG”
       DMTLOGNAME = “DMTLG_C01”

c02    GWGRP = bankg2
       TYPE = OSITP
       DOMAINID = “BA.CENTRAL01”
       DMTLOGDEV = “/usr/apps/bank/DMTLOG”
       DMTLOGNAME = “DMTLG_C02”
       URCH = “ABCD”
#
*DM_REMOTE_DOMAINS
#<remote domain name>  <domain type> <domain id>
#
b01    TYPE = TDOMAIN
       DOMAINID = “BA.BANK01”

b02    TYPE = TDOMAIN
       DOMAINID = “BA.BANK02”

b03    TYPE = TDOMAIN
       DOMAINID = “BA.BANK03”
```

```
b04    TYPE = OSITP
       DOMAINID = “BA.BANK04”
       URCH = “ABCD”

*DM_TDOMAIN
#
# <local or remote domain name>   <network address>
#
# Local network addresses
c01    NWADDR = “0x0002ff98c00b9d6d”
c01    NWADDR = “newyork01.65432”
# Remote network addresses
b01    NWADDR = “0x00020401c00b6d05”
b02    NWADDR = “dallas.65432”
b03    NWADDR = “0x00021094c00b6d9c”

*DM_OSITP
#
#<local or remote domain name> <apt> <aeq>
#                              [<aet>] [<acn>] [<apid>] [<aeid>]
#                              [<profile>]
#
c02    APT = “BA.CENTRAL01”
       AEQ = “Tuxedo.R.4.2.1”
       AET = “{1.3.15.0.3},{1}”
       ACN = “XATMI”

b04    APT = “BA.BANK04”
       AEQ = “Tuxedo.R.4.2.1”
       AET = “{1.3.15.0.4},{1}”
       ACN = “XATMI”

*DM_LOCAL_SERVICES
#<service_name>  [<Local Domain name>] [<access control>]
[<exported svcname>]
#                [<inbuftype>] [<outbuftype>]
#
open_act    ACL = branch
close_act   ACL = branch
credit
debit
balance
loan               LDOM = c02      ACL = loans

*DM_REMOTE_SERVICES
#<service_name>   [<Remote domain name>] [<local domain name>]
#              [<remote svcname>] [<routing>] [<conv>] [<trantime>]
#                 [<inbuftype>] [<outbuftype>]
#
tlr_add   LDOM = c01  ROUTING = ACCOUNT
```

```
tlr_bal   LDOM = c01  ROUTING = ACCOUNT
tlr_add   RDOM = b04  LDOM = c02 RNAME =”TPSU002”
tlr_bal   RDOM = b04  LDOM = c02 RNAME =”TPSU003”

*DM_ROUTING
# <routing criteria> <field> <typed buffer> <ranges>
#
ACCOUNT FIELD = branchid  BUFTYPE =”VIEW:account”
RANGES =”MIN - 1000:b01, 1001-3000:b02, *:b03”

*DM_ACCESS_CONTROL
#<acl name>   <Remote domain list>
#
branch  ACLIST = b01, b02, b03
loans   ACLIST = b04
```

Example 2

This example shows the Tuxedo System/Domain Configuration file required at one of the Bank Branches (BANK01).

```
#
#Tuxedo DOMAIN CONFIGURATION FILE FOR A BANK BRANCH
#
#
*DM_LOCAL_DOMAINS
#
b01    GWGRP = auth
       TYPE = TDOMAIN
       DOMAINID = "BA.BANK01"
       DMTLOGDEV = "/usr/apps/bank/DMTLOG"

*DM_REMOTE_DOMAINS
#
c01    TYPE = TDOMAIN
       DOMAINID = “BA.CENTRAL01”

*DM_TDOMAIN
#
b01    NWADDR = “0x00021094c00b689c”
c01    NWADDR = “0x0002ff98c00b9d6d”
*DM_LOCAL_SERVICES
#
tlr_add      ACL = central
tlr_bal       ACL = central
```

```
*DM_REMOTE_SERVICES
#
OPA001       RNAME = “open_act”
CLA001       RNAME = “close_act”
CRD001       RNAME = “credit”
DBT001       RNAME = “debit”
BAL001       RNAME = “balance”

DM_ACCESS_CONTROL
#
central       ACLIST = c01
```

Example 3

This example shows the configuration file entries for an eAM application:

```
#================================================================
# DMCONFIG
#     Application Domain Gateway Test Configuration
#
# See also
#     See $(TOP)/Makefile for more information.
#
# @(#)SNA Devel apps/simpsna DMCONFIG 1.6 98/03/03 15:35:29
# Copyright 1997, BEA Systems, Inc., all rights reserved.
#----------------------------------------------------------------

*DM_LOCAL_DOMAINS
simpsnad
       GWGRP=GROUP2
       TYPE=SNAX
       DOMAINID="simpsnad"
       BLOB_SHM_SIZE=1000000
       DMTLOGDEV=<your Tuxedo filesystem device and name for
       DMTLOG>

#example DMTLOGDEV="/home/me/bin/DMTLOG"

*DM_REMOTE_DOMAINS

SIMPSNAG TYPE=SNAX DOMAINID="SIMPSNAG"

*DM_SNACRM

simpcrm       SNACRMADDR="<your Host Socket Listen Address>"
              LDOM="simpsnad"
```

```
#example SNACRMADDR="0x00021770cfbd2b0d" INET family 0x0002 port
6000 host 207.189.43.13 or SNACRMADDR=//207.189.43.13:6000

*DM_SNASTACKS

simpstk
       SNACRM="simpcrm"
       STACKTYPE=<SNACRM Stack Library Named Token>
       LOCALLU=<Local LU definition specified in
       stack product>
       LTPNAME="*"
       STACKPARMS=<Parameters passed to Stack
       Product>

#example STACKTYPE="HP51"
#        LOCALLU="HPTEST"
#        STACKPARMS="testhp"  Name of the host machine

*DM_SNALINKS

simplk1    STACKREF="simpstk"
           RDOM="SIMPSNAG"
           LSYSID=<Connection ID of remote (CICS)
           region>
           RSYSID=<SYSID of remote (CICS) region>
           RLUNAME=<Alias of Applid for remote region>
           MODENAME=<Mode name VTAM mode entry>
           SECURITY="LOCAL"
           STARTTYPE="COLD"
           MAXSESS=<Total Session number>
           MINWIN=<Session Local Winners>
           MAXSYNCLVL=<0|1|2 Maximum Syncpoint Level>

#example LSYSID="BEA"
#        RSYSID="TEST"
#        RLUNAME="CICSTEST"
#        MODENAME="SMSNA100"
#        MAXSESS=10
#        MINWIN=5
#        MAXSYNCLVL=2

*DM_LOCAL_SERVICES

MIRROR LDOM="simpsnad"
       CONV=N
       RNAME="MIRRORSERV"
       INBUFTYPE="STRING"
       OUTBUFTYPE="STRING"
       API="ATMI"

*DM_REMOTE_SERVICES

SIMPDPL AUTOTRAN=N
       LDOM="simpsnad"
       RDOM=SIMPSNAG
       CONV=N
       RNAME="TOUPDPLS"
       INBUFTYPE="STRING"
       OUTBUFTYPE="STRING"
       API="ATMI"
       FUNCTION="DPL"

SIMPDTP AUTOTRAN=N
       LDOM="simpsnad"
       RDOM=SIMPSNAG
       CONV=N
       RNAME="DTPS"
       INBUFTYPE="STRING"
       OUTBUFTYPE="STRING"
       API="ATMI"
       FUNCTION="APPC"
```

See Also

`build_dgw`(1), `dmadmin`(1), `tmboot`(1), `tmshutdown`(1), `dmloadcf`(1), `dmunloadcf`(1)

`dmgwopts`(5), `GWADM`(5), `DMADM`(5)

Tuxedo /Domain User Guide

Tuxedo Administrator’s Guide

Tuxedo Programmer’s Guide

dmloadcf

Parse a DMCONFIG file and load binary BDMCONFIG configuration file

Synopsis

```
dmloadcf [-c] [-n] [-y] [-b blocks] {dmconfig_file | - }
```

Description

dmloadcf reads a file or the standard input that is in DMCONFIG syntax, checks the syntax, and optionally loads a binary BDMCONFIG configuration file. The BDMCONFIG environment variable points to the path name of the BDMCONFIG file where the information should be stored.

dmloadcf prints an error message if it finds any required section of the DMCONFIG file missing. If a syntax error is found while parsing the input file, dmloadcf exits without performing any updates to the BDMCONFIG file.

dmloadcf requires the existence of the $TUXDIR/udataobj/DMTYPE file. This file defines the valid domain types. If this file does not exist, *dmloadcf* exits without performing any updates to the BDMCONFIG file.

The effective user identifier of the person running dmloadcf must match the UID in the RESOURCES section of the TUXCONFIG file.

The -c option to dmloadcf causes the program to print minimum IPC resources needed for each local domain (gateway group) in this configuration. The BDMCONFIG file is not updated.

The -n option to dmloadcf causes the program to do only syntax checking of the ASCII DMCONFIG file without actually updating the BDMCONFIG file.

After syntax checking, dmloadcf checks to see if the file pointed to by BDMCONFIG exists, is a valid Tuxedo System file system, and contains BDMCONFIG tables. If these conditions are not true, the user is prompted to create and initialize the file with

```
Initialize BDMCONFIG file: path [y, q]?
```

where *path* is the complete file name of the BDMCONFIG file. Prompting is suppressed if the standard input or output are not terminals, or if the -y option is specified on the command line. Any response other than “y” or “Y” will cause dmloadcf to exit without creating the configuration file.

If the BDMCONFIG file is not properly initialized, and the user has given the go-ahead, dmloadcf creates the Tuxedo file system and then creates the BDMCONFIG tables. If the -b option is specified on the command line, its argument is used as the number of blocks for the device when creating the Tuxedo file system. If the value of the -b option is large enough to hold the new BDMCONFIG tables, dmloadcf will use the specified value to create the new file system; otherwise, dmloadcf will print an error message and exit. If the -b option is not specified, dmloadcf will create a new file system large enough to hold the BDMCONFIG tables. The -b option is ignored if the file system already exists. The -b option is highly recommended if BDMCONFIG is a raw device (that has not been initialized) and should be set to the number of blocks on the raw device. The -b option is not recommended if BDMCONFIG is a regular UNIX file.

If the BDMCONFIG file is determined to already have been initialized, dmloadcf ensures that the local domain described by that BDMCONFIG file is not running. If a local domain is running, dmloadcf prints an error message and exits. Otherwise, dmloadcf, to confirm that the file should be overwritten, prompts the user with:

```
“Really overwrite BDMCONFIG file [y, q]?”
```

Prompting is suppressed if the standard input or output are not a terminal or if the -y option is specified on the command line. Any response other than “y” or “Y” will cause dmloadcf to exit without overwriting the file.

If the SECURITY parameter is specified in the RESOURCES section of the TUXCONFIG file, then dmloadcf will flush the standard input, turn off terminal echo and prompt the user for an application password as follows:

```
Enter Application Password?
```

The password is truncated to 8 characters. The option to load the ASCII DMCONFIG file via the standard input (rather than a file) cannot be used when this SECURITY parameter is turned on. If the standard input is not a terminal, that is, if the user cannot be prompted for a password (as with a here file, for example), then the environment variable APP_PW is accessed to set the application password. If the environment variable APP_PW is not set with the standard input not a terminal, then dmloadcf will print an error message, generate a log message and fail to load the BDMCONFIG file.

Assuming no errors, and if all checks have passed, `dmloadcf` loads the `DMCONFIG` file into the `BDMCONFIG` file. It will overwrite all existing information found in the `BDMCONFIG` tables.

Portability

This command is supported as a Tuxedo-supplied administrative tool on UNIX and Windows NT operating systems.

Environment Variables

The environment variable `APP_PW` must be set for applications that require security (the `SECURITY` parameter in the `TUXCONFIG` file is set to `APP_PW`) and `dmloadcf` is run with something other than a terminal as the standard input.

The `BDMCONFIG` environment variable should point to the BDMCONFIG file.

Examples

The following example shows how a binary configuration file is loaded from the `bank.dmconfig` ASCII file. The `BDMCONFIG` device is created (or re-initialized) with 2000 blocks:

```
dmloadcf -b 2000 -y bank.dmconfig
```

Diagnostics

If an error is detected in the input, the offending line is printed to standard error along with a message indicating the problem. If a syntax error is found in the `DMCONFIG` file or the system is currently running, no information is updated in the `BDMCONFIG` file and `dmloadcf` exits with exit code 1.

If `dmloadcf` is run on an active node, the following error message is displayed:

```
*** dmloadcf cannot run on an active node ***
```

If `dmloadcf` is run by a person whose effective user identifier doesn't match the `UID` specified in the `TUXCONFIG` file, the following error message is displayed:

```
*** UID is not effective user ID ***
```

Upon successful completion, `dmloadcf` exits with exit code 0. If the `BDMCONFIG` file is updated, a `userlog` message is generated to record this event.

See Also

`dmunloadcf`(1), `dmconfig`(5), `ubbconfig`(5)

Tuxedo /Domain User Guide

Tuxedo Administrator's Guide

dmunloadcf

Unload binary `BDMCONFIG` domain configuration file

Synopsis

```
dmunloadcf
```

Description

`dmunloadcf` translates the `BDMCONFIG` configuration file from the binary representation into ASCII. This translation is useful for transporting the file in a compact way between machines with different byte ordering and backing up a copy of the file in a compact form for reliability. The ASCII format is the same as is described in `dmconfig`(5).

`dmunloadcf` reads values from the `BDMCONFIG` file pointed to by the `BDMCONFIG` environment variable and writes them to its standard output.

Portability

This command is supported as a Tuxedo-supplied administrative tool on UNIX and Windows NT operating systems.

Examples

To unload the configuration in `/usr/tuxedo/BDMCONFIG` into the file `bdmconfig.backup`:

```
BDMCONFIG=/usr/tuxedo/BDMCONFIG dmunloadcf > bdmconfig.backup
```

Diagnostics

`dmunloadcf` checks that the file pointed to by the `BDMCONFIG` environment variable exists, is a valid Tuxedo file system, and contains `BDMCONFIG` tables. If any of these conditions is not met, `dmunloadcf` prints an error message and exits with error code 1. Upon successful completion, `dmunloadcf` exits with exit code 0.

See Also

`dmloadcf`(1), `dmconfig`(5)

Tuxedo /Domain User Guide

GWADM

/Domain gateway administrative server.

Synopsis

```
GWADM SRVGRP = “identifier” SRVID = “number” REPLYQ = “N”
      CLOPT = “-A -- [-a { on | off } ] [-s services ]
      [-t { on | off } ]“
```

Description

The gateway administrative server (`GWADM`) is a Tuxedo-supplied server that provides administrative functions for a /Domain gateway group.

GWADM should be defined in the SERVERS section of the UBBCONFIG file as a server running within a particular gateway group, that is, SRVGRP must be set to the corresponding GRPNAME tag specified in the GROUPS section. The SVRID parameter is also required and its value must consider the maximum number of gateways allowed within the gateway group.

There should be only one instance of a GWADM per /Domain gateway group, and it should NOT be part of the MSSQ defined for the gateways associated with the group. Also, GWADM should have the REPLYQ attribute set to N.

The CLOPT option is a string of command line options that is passed to the GWADM when it is booted. This string has the following format:

```
CLOPT=”-A -- <gateway group runtime parameters>”
```

The following runtime parameters are recognized for a gateway group:

`-a {` *`on`* `|` *`off`* `}`
: This option turns `off` or `on` the audit log feature for this local domain. The default is `off`. The `dmadmin` program can be used to change this setting while the gateway group is running (see `dmadmin`(1)).

`-s` *`services`*

Specifies the remote *`services`* that should be initially offered by the domain gateway. The specifications for these services are found in the DMCONFIG file. For example, the specification

```
-s x,y,z
```

implies that the gateway should initially advertise remote services `x`, `y`, and `z`. Spaces are not allowed between commas and the `-s` option may appear several times.

`-t {` *`on`* `|` *`off`* `}`

This option turns `off` or `on` the statistics gathering feature for the local domain. The default is `off`. The `dmadmin` program can be used to change this setting while the gateway group is running (see `dmadmin`(1)).

The GWADM server must be booted before the corresponding gateways.

Portability

This server is supported on Tuxedo-supplied servers, using UNIX System and Windows NT operating systems.

Interoperability

The initial release of SNA-type gateways can only be installed on a node running Tuxedo Release 6.5.

Examples

The following example illustrates the definition of the administrative server in the UBBCONFIG file.

```
#
*GROUPS
DMADMGRP  GRPNO=1
gwgrp    GRPNO=2
```

```
#
*SERVERS
DMADM SRVGRP=”DMADMGRP” SRVID=1001 REPLYQ=N RESTART=Y GRACE=0
GWADM SRVGRP=”gwgrp” SRVID=1002 REPLYQ=N RESTART=Y GRACE=0
      CLOPT=”-A -- -a on -t on”
SNACRM SRVGRP=”gwgrp” SRVID=1003 CLOPT=”-A--//host:6000 gwgrp”
GWSNAX SRVGRP=”gwgrp” SRVID=1004 RQADDR=”gwgrp” REPLYQ=N
```

See Also

`dmadmin`(1), `tmboot`(1)

`dmconfig`(5), `DMADM`(5), `servopts`(5), `ubbconfig`(5)

Tuxedo /Domain User Guide

Tuxedo Administrator’s Guide

GWSNAX

This is the gateway server process for eAM.

Synopsis

```
GWSNAX SRVGRP = “identifier” SRVID = “number” REPLYQ = “N”
     CLOPT = “-A -- [-e {type} -m -n {type:min:max} -t {number}
     -u {keyfile}]“
```

Description

The GWSNAX server provides Tuxedo functions for an eAM gateway group.

GWSNAX should be defined in the SERVERS section of the UBBCONFIG file as a server running within a particular gateway group; that is, SRVGRP must be set to the corresponding GRPNAME tag specified in the GROUPS section. The SVRID parameter is also required and its value must consider the maximum number of gateways allowed within the gateway group. The GWSNAX definition must not precede its associated CRM server definition in the UBBCONFIG file.

There should be only one instance of a GWSNAX per eAM gateway group, and it should NOT be part of the MSSQ defined for the gateways associated with the group.

The CLOPT option is a string of command line options that is passed to the GWSNAX when it is booted. This string has the following format:

```
CLOPT=”-A -- <gateway runtime parameters>”
```

The following runtime parameters are recognized for a gateway:

`-e {type}`
: This option specifies that data transformation is in effect for this process. Using `ii` value indicates eLink Information Integrator is used.

`-m`
: This option specifies that userid mapping is bypassed. Any userid mapping already defined in the DMCONFIG file is preserved, but is not in effect.

`-n {type:min:max}`

Establishes that encryption is in effect for this client process. `type` is the encryption type. Currently, the only valid entry is GPE.The min and max values designate the minimum and maximum number of bits to be used for encryption. This level is used during the negotiation between the CRM and client process. Any number is acceptable, but the negotiated values resolve to 0, 56, or 128.

`min` designates the minimum number of bits to be used for encryption. This level is used during the negotiation between the CRM and GWSNAX. Any number is acceptable, but the negotiated values resolve to 0, 56, or 128. The level specified must be supported by the security add-on package used. `max` designates the maximum number of bits to be used for encryption. This level is used during the negotiation between the CRM and GWSNAX. Any number is acceptable, but the negotiated values resolve to 0, 56, or 128.

`-t {`*`number`*`}`

This option indicates the type of character string transformation the gateway performs. (Refer to Table A-10 for values.)

`-u {keyfile}`

Establishes that process authentication is in effect for communications between this process and the CRM.
The `keyfile` is the location file containing a hash key known to both this process and the CRM. The file contains a single line specifying a unique hash key (limited to eight characters). The file should be protected.

Table A-10 C to COBOL String Transformation

CLOPT -t Parameter Value	Tuxedo Application Language	Host Application Language
Not Set	No string transformation established	
1	C	COBOL
2	COBOL	C
3	C	C
4	COBOL	COBOL

Portability

Refer to Table 1-1 for a complete listing of compatible operating systems.

Interoperability

Refer to Table 1-1 for a complete listing of supported platforms.

Examples

The following example illustrates the definition of the administrative server in the UBBCONFIG file.

```
#
*GROUPS
DMADMGRP GRPNO=1
gwgrp    GRPNO=2
#
*SERVERS
DMADM SRVGRP=”DMADMGRP” SRVID=1001 REPLYQ=N RESTART=Y GRACE=0
GWADM SRVGRP=”gwgrp” SRVID=1002 REPLYQ=N RESTART=Y GRACE=0
      CLOPT=”-A -- -a on -t on”
SNACRM SRVGRP=”gwgrp” SRVID=1003 CLOPT=”-A--//host:6000 gwgrp”
GWSNAX SRVGRP=”gwgrp” SRVID=1004 RQADDR=”gwgrp” REPLYQ=N
       CLOPT=”-- -t 1”
```

See Also

`dmadmin`(1), `tmboot`(1)

`dmconfig`(5), `DMADM`(5), `servopts`(5), `ubbconfig`(5)

Tuxedo /Domain User Guide

Tuxedo Administrator’s Guide

modusr

Modify a remote user password.

Synopsis

```
modusr -d <local domain> ID -R <remote domain ID> -u <remote
username>
```

Description

`modusr` can only be executed as a subcommand of `dmadmin`(1). The purpose of this page is to describe options for the subcommand and to show an example.

The subcommand allows the administrator to modify passwords in the remote password table. The administrator is prompted for the remote password.

The table entries modified are used for passing remote user names and passwords to remote SNA domains when the application is using SNA-type gateways and `SECURITY` is set to `USER_AUTH`, `ACL`, or `MANDATORY_ACL` in the `ubbconfig` file and `SECURITY` is set to `DM_USER_PW` in the `DMCONFIG` file.

The following options are available:

`-d` *<local domain ID>*
: This is the name of the local domain gateway with which the ids and passwords are associated. This is the same ID as the one used when creating the domain definitions either in the `DMCONFIG` file or through the Graphical Administrative Interface.

`-R` *<remote domain ID>*
: This is the name of the remote domain gateway with which the ids and passwords are associated. This is the same ID as the one used when creating the domain definitions either in the `DMCONFIG` file or through the Graphical Administrative Interface.

`-u <`*`remote username`*`>`

The remote user whose password is being modified.

Before running this subcommand the application must be configured using either the Graphical Administrative Interface or `tmloadcf`(1) and `dmloadcf`(1). `dmadmin modusr` may be run on any active node.

Portability

This subcommand is available on the latest version of Tuxedo, as documented for this release of BEA eLink Adapter for Mainframe.

Diagnostics

The `dmadmin modusr` subcommand exits with a return code of 0 upon successful completion.

Examples

```
modusr -d tux -R cics -u CICSUSR /*modifies remote user’s password
                                  sent to CICS. The administrator
                                 is prompted for the password*/
```

See Also

`delusr`(5), `addusr`(5)

SNACRM

Launches the Communications Resource Manager.

Synopsis

```
SNACRM [ -t 0|1|2|3 ]  [-s] [-o]] [-n <type>:min:max] [-u <keyfile>]
       [-p<  > <addr> <group>]
```

Description

SNACRM communicates between an eAM gateway and a remote application using SNA protocol.

On a UNIX platform, you can manually start SNACRM from the command line or as an ATMI server. On a MicroSoft NT platform, you must start the CRM as an ATMI server.

Note: In most cases, it is recommended that you start the CRM as a Tuxedo server. Please refer to “Specifying the SNACRM as a BEA Tuxedo Server” in the section “Configuring the System.”

When you start SNACRM from the UNIX command line, the CRM Command Line Console puts its prompt in the window, and if exited, shuts down all of the active links. When started from DMINIT, the console is redirected to the null device.

When using TMADMIN to start and stop servers by group ID, include the DMINIT server in the same group so that CRM can be restarted with its corresponding SNA Domain Gateway.

You must configure one CRM for each eAM gateway, as well as configuring one stack for each CRM definition. Each stack can manage one or more SNA links, which is equivalent to a Tuxedo remote domain. The CRM must be started and listening before the eAM gateway is booted.

`SNACRM` has two types of log files stored in `$APPDIR, RSTRTLOG and BLOBLOG.` `RSTRTLOG` is the transaction state log used during the recovery process, while the `BLOBLOG` log stores session and link information. Deleting the log files require a cold start for each link involved. You can use the `CRMLOGS` command to display the contents and state of the `SNARCM` log files.

Trace Options

When initiating the CRM from the UNIX command line or via the DMINIT server, you can specify any of the following trace levels:

- *`0`* = No tracing. Setting this level effectively disables `SNACRM` tracing and closes the trace file, if there is one. If tracing is subsequently restarted, a new file is created with an incremental numerical suffix.
- *`1`* = Minimum tracing. At this level, `SNACRM` traces only major events and is sufficient only to determine the sequence of application conversations.
- *`2`* = Medium tracing. At this level, `SNACRM` also traces all I/O buffers.
- *`3`* = Maximum tracing. At this level, `SNACRM` also traces all APPC verbs.

The APPC Protocol Stack API trace is either enabled or disabled. If enabled, it generally shows the parameters and results of all API calls. Depending on the stack being used, other options (such as vendor-specified environment variables) may have to be activated for `SNACRM` to enable the trace.

Note: When specifying the CRM as a server on a MicroSoft NT platform, you can establish the above trace levels in the `CLOPT` parameter for the server definition in the UBBCONFIG file. Refer to BEA TUXEDO publications for information about the `CLOPT` parameter.

General Options

The following parameters apply to this command:

`-s`

APPC Stack API trace (default none).

Note: On OS/390 platforms, the General Trace Facility (GTF) is used to capture API trace records under user EID 2EA. The GTF must be active on the OS/390 platform to use the `-s` parameter.

`-n`
: Establishes that encryption is in effect for this process.

`type`
: Is the encryption type. Currently, the only valid entry is GPE (for general purpose encryption).

`min`
: Designates the minimum number of bits to be used for encryption. These levels are used during the negotiation between the CRM and client process. Any number is acceptable, but the negotiated values resolve to 0, 56, or 128.

`max`
: Designates the maximum number of bits to be used for encryption. These levels are used during the negotiation between the CRM and client process. Any number is acceptable, but the negotiated values resolve to 0, 56, or 128.

`-u <keyfile>`
: Establishes that process authentication is in effect for communications between the CRM and a client process.

  `keyfile` is the location of a file containing a hash key known to both the CRM and the client process. The file contains a single line specifying a unique hash key (limited to eight characters). The file should be protected.

HexSocketAddress
: Is a TCP/IP address using //hostname:port_addr or the `sockaddr_in` format of family, port, address:
  `<0xFFFFPPPPAAAAAAAA>`
  In this entry, arguments and options are defined in the following way:
  `FFFF` is the hex value of the protocol family, always `0x0002` for the INET family.
  `PPPP` is the hex value of an unused TCP/IP port.
  `AAAAAAAA` is the hex value of the IP address for the machine running the `SNACRM`.

`group`
: eAM Gateway Group Name (required)

Performance Option

The performance option is a tunable parameter used only for the OS/390 Unix and MVS SNACRM. These SNACRM versions use threads to process a request and this parameter indicates how many threads to start for processing SNA requests.

The performance option uses the following format:

```
-p <nbr>
```

In this argument, <nbr> equals the number of threads to start. The default for this value is 100 threads. This value should correspond to the load of SNA requests that will be made concurrently. If the number of requests exceeds the number of threads, the request will still be executed, however, the completion time could be affected. Do not exceed 200 threads. The SNACRM is tuned for a maximum of 200 threads. Lower the 100 threads value if you have a restriction on the number of threads that can be active in your system.

Notes: Do not use the `-o` and `-a` parameters for the `SNACRM` command.

If the CRM is running under MVS, the `-u` option should be specified as: `-u DD:ddname.` In this argument, `ddname` is a 1 to 8 byte `DD` statement that will identify the dataset name in the JCL.

Environment Variables

You must set the following environment variables before starting the CRM:

- From the UNIX command line:

```
FIEDLTBLS32 must contain fmb.def.
FLDTBLDIR32 must contain the path to the eAM libraries directory
(for example, eAM40/lib).
APPDIR must be set to the application directory.
```

- From MVS:

```
FLDTBLDIR32=DD:FLDTBL
FIELDTBLS32=FMB
APPDIR=<High level qualifier for datasets to be created in
APPDIR>
```

Portability

Refer to the *BEA eLink Adapter for Mainframe Release Notes* for a list of operating systems and stacks supporting CRM.

Interoperability

Refer to the *BEA eLink Adapter for Mainframe Release Notes* for information about interoperability.

Diagnostics

`SNACRM` exits with a return code of 0 upon successful completion.

Examples

Following is an example of the `SNACRM` command:

```
SNACRM 0 //myhost:5587 GROUP2 /dev/null>std.out 2>std.err &
```

When you start `SNACRM` from the UNIX command line, the following `SNARCM` command Line Console appears:

```
$ SNACRM -t 0 //myhost:5587 GROUP2
       BEA eLink SNA Resource Manager started Thu Dec 11
       18:40:49.098 1997
       [SNACRM]

       Console active. Enter commands
       ?>
        da => Display active tasks
        dl => Display remote links
        ds => Display link statistics
        dt => Display trace status
        st => Start all links
        sh => Stop all links and terminate
        si => Terminate immediately (no quiesce)
```

To launch `SNACRM` with the console running in the background:

```
$ SNACRM -t0 //myhost:5587 GROUP2 <dev/null>std.out 2>std.err &
```

To launch `SNARCM` with detailed tracing and APPC Stack API tracing turned on from the command line using the host/port address, type:

```
SNARCM -t2 -s //myhost:5587
```

When using the `DMINIT` server to launch the CRM, you must specify the `CLOPT` option in `UBCONFIG.CFG` as follows:

```
CLOPT=”-- -f filename”
```

Where the filename is the name of a shell script containing the start-up command line for one or more CRM processes.

See Also

`dmadmin`, `dmconfig`, and `xsnacrm`

xsnacrm

X/Motif real-time monitor for running the CRM (not available on Windows NT platforms)

Synopsis

```
xsnacrm [ X overrides ]  address  [ address  . . . ]
```

(See syntax examples.)

Description

The `xsnacrm` program provides real-time monitoring of running CRMs and displays information describing the activity occurring in each CRM. The `xsnacrm` utility is intended to be used by administrators and system operators only. Therefore, usage may be restricted by the installation (by setting the execute permissions). `xsnacrm` requires Motif libraries.

Note: The `xsnacrm` utility cannot be executed on an OS/390 platform.

Command Line Options

`xsnacrm` supports the standard X Toolkit command line arguments (see X(1)). The following additional arguments are supported as well.

General Options

The following general option is available:

`address`

Specifies the host name and port number of a CRM to monitor. This value must match the corresponding parameter on the command line used to start the CRM you wish to monitor.

There must be at least one address specified. Any number of CRMs may be monitored by specifying all their associated addresses.

The format of an address consists of two forward slashes (//) followed by a host name or address, a colon (:), and a service name or decimal port number (containing no spaces).

If a host name is used, it should be an entry in the file /etc/hosts. If a host address is used, it should be specified in the format nnn.nnn.nnn.nnn where each group of nnn represents a decimal number between 1 and 255. This host should identify the computer where the CRM you wish to monitor is running, ***not*** the host where `xsnacrm` is to run.

If a service name is used, it should be an entry in the file /etc/services. If a decimal port number is used, it should be a decimal number in the range 4000 - 32767. This number must match the corresponding port number on the command line used to start the CRM you want to monitor. (If the CRM was started automatically, the address is specified in the DMCONFIG file).

xsnacrm Window

`xsnacrm` displays a single window consisting of the following sections from top to bottom:

- Title Frame

  Displays the application title "*BEA eLink SNA CRM Status*"

- Menu Bar

Displays the menu items “File” and “Trace.” The File menu consists of a single “Exit” button that terminates `xsnacrm`. The `xsnacrm` window may also be terminated by selecting “close” on the X/Motif system menu for the window.

The Trace menu contains two sections that send commands to the currently selected CRM to change it’s own tracing function, and the tracing function of the APPC Protocol Stack the CRM is using, respectively. To change either current tracing option, select the corresponding menu button (For more information on tracing, please refer to the “Trace Options” section).

- BEA Logo

  Displays the BEA Logo.

- CRM Select Pane

  Displays the list of CRMs specified on the command line. The list consists of a set of radio buttons. The selected button determines which CRM’s data is displayed in the other panes below.

  The phrase “**not active or invalid address**” means that `xsnacrm` is unable to connect to the INET address specified, because the:

  - Address is incorrect
  - CRM is not monitoring the address (probably because it is not running)
  - Path to CRM is not available (perhaps due to a network problem)

- Trace Status Pane

  Displays the current trace options for the selected CRM.

  You can enter one of the following `xsnacrm` trace levels:

  - *`0`* = No tracing. Setting this level effectively disables `SNACRM` tracing and closes the trace file if there is one. If tracing is subsequently restarted, a new file will be created with an incremental numerical suffix.
  - *`1`* = Minimum tracing. At this level, the `SNACRM` traces only major events and is sufficient only to determine the sequence of application conversations.
  - *`2`* = Medium tracing. At this level, `SNACRM` also traces all I/O buffers.
  - *`3`* = Maximum tracing. At this level, `SNACRM` also traces all APPC verbs.

  The APPC Protocol Stack API trace is either **enabled** or **disabled**. If enabled, it generally shows the parameters and results of all API calls. Depending on the

Stack being used, other options (such as vendor-specified environment variables) may have to be activated for `SNACRM` to enable the trace.

- Link Status Pane

  Displays the current status of all remote links for the selected CRM. The text may be scrolled if it is not entirely visible. Possible status values are:

  ```
  Ins = The link is In Service
  Acq = The link connection has been acquired
  Xok = The link connection has been acquired and is OK
  ```

- Link Statistics Pane

  Displays the current statistics for all remote links for the selected CRM. The text may be scrolled if it is not entirely visible.

- Message Line

  Displays messages showing the results of either automatic attempts by `xsnacrm` to connect to the specified CRMs or commands issued to change the trace options.

  The space in the window allocated to each of the four panes can be adjusted by dragging the sashes (little rectangles) located on the dividers between them.

Examples

The default geometry for `xsnacrm` is 630x480+150+150. This places an appropriately sized window for the default font in approximately the center of a 1024x768 Xterm. The following command places this window in the lower-right corner at start-up:

```
xsnacrm -geometry 630x480-0-0 //somehost:4999 //otherhost:6666
```

The following command starts `xsnacrm` as an icon:

```
xsnacrm -iconic //252.148.37.16:5555
```

The following command changes the name of the trace menu to *Commands* and uses the service name *snacrm* for the port number:

```
xsnacrm -xrm “*tracemenu.labelString: Commands” //myhost:snacrm
```

Customizing X Resources

The default X resources for `xsnacrm` correspond to the distributed contents of the associated file xsnacrm. To customize the application, copy the xsnacrm file to your home directory and edit it.

Widgets

The widget structure of the xsnacrm window is given in the text of the xsnacrm file as follows:

```
! English US resource file for xsnacrm program
!
!   "@(#)ISC Devel SNACRM Xsnacrm 1.1 97/08/12 17:49:57”;
!
! The values shown below are the fallback resource values
!
! The widget hierarchy is:
!
!  Xsnacrm               App Shell
!   mainWindow           Main Window
!    logo                Frame
!     logobitmap         Label
!    menubar             Row/Column
!     filemenu           Pull-down Menu
!      quit              Push Button
!     tracemenu          Pull-down Menu
!      tracebutton0      Push Button
!      tracebutton1      Push Button
!      tracebutton2      Push Button
!      tracebutton3      Push Button
!      traceSep          Separator
!      tracebuttonY      Push Button
!      tracebuttonN      Push Button
!    mainpane            Paned Window
!     selectFrame        Frame
!      selectFrameLabel Label
!      selectRadioBox    Row/Column
!       selectButton<n> Toggle Button
!     traceFrame         Frame
!      traceFrameLabel  Label
!      traceData         Label
!     statusFrame        Frame
!      statusFrameLabel Label
!      stusScroll       Scrolled Window
!       stusScrollData  Label
!     statisticsFrame   Frame
!      statFrameLabel   Label
!      statScroll       Scrolled Window
!       statScrollData  Label
!    mainmessage        Label
!   quitDialog          Message Dialog
!

*title:                 BEA eLink SNA CRM Status
*geometry:              630x480+150+150
*foreground:            white
*background:            purple
*fontList:              *courier-medium-r-normal--12*
*filemenu.labelString:  File
*quitDialog.okLabelString: Exit
*quitDialog.messageString: Exit SNA CRM Status Display now?
*quit.labelString:         Exit
*tracemenu.labelString:    Trace
*traceButton0.labelString: Stop CRM Trace
*traceButton1.labelString: Set Minimum CRM Trace
*traceButton2.labelString: Set Medium CRM Trace
*traceButton3.labelString: Set Maximum CRM Trace
*traceButtonY.labelString: Start APPC Stack Trace
*traceButtonN.labelString: Stop APPC Stack Trace
```

See Also

`xsnacrm` and `SNACRM`

B Error Messages

The following messages are logged to the USERLOG file from an eAM gateway (GWSNAX) and the Communications Resource Manager (CRM).

1118:INFO	**A BLOCKING timeout has occurred on an outbound request**	
	DESCRIPTION	A blocking time out has occurred on a request to the host.
	ACTION	Examine the ULOG for more information concerning the reason for the failure. Check configuration and increase the BLOCKTIME value, if needed.
1121:ERROR	**Bad action state**	
	DESCRIPTION	An internal error has been detected processing the action.
	ACTION	Contact BEA Customer Support.
1122:ERROR	**Invalid action index**	
	DESCRIPTION	An internal error has been detected, gw_nw_acall has returned a failure.
	ACTION	Contact BEA Customer Support.
1600:ERROR	**Invalid action index**	
	DESCRIPTION	An internal error has been detected, gw_nw_connect has returned a failure.
	ACTION	Contact BEA Customer Support.
1602:ERROR	**Cannot open SNA connection**	

	DESCRIPTION	Unable to allocate a network context. gw_nw_AllocNwCtx failed.
	ACTION	See previously printed ULOG message for cause.
1604:ERROR	**Initialization of environment failed**	
	DESCRIPTION	Unable to initialize the conversation environment.
	ACTION	See previously printed ULOG message for cause.
1621:ERROR	**Bad action state**	
	DESCRIPTION	An internal error has been detected processing the action.
	ACTION	Contact BEA Customer Support.
1622:ERROR	**Invalid action index**	
	DESCRIPTION	An internal error has been detected, gw_nw_convsend has returned a failure.
	ACTION	Contact BEA Customer Support.
1625:ERROR	**Invalid action index**	
	DESCRIPTION	An internal error has been detected, gw_nw_disconnect has returned a failure.
	ACTION	Contact BEA Customer Support.
1630:ERROR	**Unable to send data on sna_conv_idx %d**	
	DESCRIPTION	A network error has occurred while processing a send to the CRM. gw_nw_convsend failed.
	ACTION	Determine if CRM is still active. Look for other ULOG messages that specify a reason for this failure.
1989:ERROR	**External encode/decode service returned error: \n\tTUXEDO code (%d) %s**	
	DESCRIPTION	A Tuxedo error has occurred in the Data Integration portion of the product.

	ACTION	Use the Tuxedo error to resolve the Data Integration problem. If unable to resolve the problem, contact BEA Customer Support.
1990:ERROR	**Cannot create external decode action!**	
	DESCRIPTION	An internal error has occurred in the Data Integration portion of the product.
	ACTION	Contact BEA Customer Support.
1991:ERROR	**Cannot create external encode action!**	
	DESCRIPTION	An internal error has occurred in the Data Integration portion of the product.
	ACTION	Contact BEA Customer Support.
1992:ERROR	**Invalid action index**	
	DESCRIPTION	An internal error has been detected, gw_nw_decode has returned a failure.
	ACTION	Contact BEA Customer Support.
1993:ERROR	**Invalid action index**	
	DESCRIPTION	An internal error has been detected, gw_nw_encode has returned a failure.
	ACTION	Contact BEA Customer Support.
1994:ERROR	**Context data incorrect - null context index**	
	DESCRIPTION	An internal error has been detected, gw_nw_decode has returned a failure.
	ACTION	Contact BEA Customer Support.
1995:ERROR	**Context data incorrect - null context index**	
	DESCRIPTION	An internal error has been detected, gw_nw_encode has returned a failure.
	ACTION	Contact BEA Customer Support.
2200:ERROR	**Cannot malloc sna structures**	

	DESCRIPTION	An internal error has been detected, gw_nw_init was unable to allocate memory needed.
	ACTION	Contact BEA Customer Support.
2203:ERROR	**Cannot malloc remote domain structure**	
	DESCRIPTION	An internal error has been detected, gw_nw_init was unable to allocate memory needed.
	ACTION	Contact BEA Customer Support.
2204:ERROR	**Cannot malloc local SNA structure**	
	DESCRIPTION	An internal error has been detected, gw_nw_init was unable to allocate memory needed.
	ACTION	Contact BEA Customer Support.
2205:ERROR	**Unable to obtain local snadom information from shmem**	
	DESCRIPTION	An internal error has been detected, gw_nw_init was unable to get needed information from shared memory.
	ACTION	Contact BEA Customer Support.
2206:ERROR	**Cannot malloc remote service structure**	
	DESCRIPTION	An internal error has been detected, gw_nw_init was unable to allocate memory needed.
	ACTION	Contact BEA Customer Support.
2207:ERROR	**Cannot malloc remote SNA structure**	
	DESCRIPTION	An internal error has been detected, gw_nw_init was unable to allocate memory needed.
	ACTION	Contact BEA Customer Support.
2210:ERROR	**Can’t create listener for this GATEWAY**	
	DESCRIPTION	An internal error has been detected, gw_nw_init was unable to create a listener.
	ACTION	Contact BEA Customer Support.

2211:WARNING	**NO SNA conversations waiting for startup**	
	DESCRIPTION	No conversations are waiting at startup.
	ACTION	No action needs to be taken.
2214:ERROR	**Unable to set APPC_GATEWAY environment variable**	
	DESCRIPTION	An internal error has been detected, _gp_get_nodename failed.
	ACTION	Contact BEA Customer Support.
2220:ERROR	**Memory Allocation Failure**	
	DESCRIPTION	An internal error has been detected, gw_nw_init was unable to allocate memory needed.
	ACTION	Contact BEA Customer Support.
2221:ERROR	**Memory Allocation Failure**	
	DESCRIPTION	An internal error has been detected, gw_nw_init was unable to allocate memory needed.
	ACTION	Contact BEA Customer Support.
2222:ERROR	**Memory Allocation Failure for RDOM table**	
	DESCRIPTION	An internal error has been detected, gw_nw_init was unable to allocate memory needed.
	ACTION	Contact BEA Customer Support.
2223:ERROR	**Memory Allocation Failure for SNALINK table**	
	DESCRIPTION	An internal error has been detected, gw_nw_init was unable to allocate memory needed.
	ACTION	Contact BEA Customer Support.
2224:ERROR	**Memory Allocation Failure for SNACRM table**	
	DESCRIPTION	An internal error has been detected, gw_nw_init was unable to allocate memory needed.
	ACTION	Contact BEA Customer Support.

2225:ERROR	**Memory Allocation Failure for SNASTACK table**	
	DESCRIPTION	An internal error has been detected, gw_nw_init was unable to allocate memory needed.
	ACTION	Contact BEA Customer Support.
2226:ERROR	**Unable to access SNALINK information from memory**	
	DESCRIPTION	An internal error has been detected, gw_nw_init was unable to get needed information from shared memory.
	ACTION	Contact BEA Customer Support.
2227:ERROR	**Unable to access SNACRM information from memory**	
	DESCRIPTION	An internal error has been detected, gw_nw_init was unable to get needed information from shared memory.
	ACTION	Contact BEA Customer Support.
2228:ERROR	**Unable to convert SNACRM network address for %s**	
	DESCRIPTION	An error has occurred while trying to contact the SNACRM during startup.
	ACTION	Examine the ULOG for more information concerning the reason for the failure. Probably due to a configuration problem.
2229:ERROR	**Unable to access SNASTACK information from memory**	
	DESCRIPTION	An internal error has been detected, gw_nw_init was unable to get needed information from shared memory.
	ACTION	Contact BEA Customer Support.

2230:ERROR	Unable to build internal network tables	
	DESCRIPTION	An internal error has been detected, gw_nw_build_nwtbls failed.
	ACTION	Examine the ULOG for more information concerning the reason for the failure. Probably due to a configuration problem.
2231:ERROR	**Unable to connect to SNACRM %s**	
	DESCRIPTION	An error has occurred while trying to contact the CRM during startup.
	ACTION	Examine the ULOG for more information concerning the reason for the failure. Possible reasons are: 1. The CRM server was not configured in the UBBCONFIG. 2. The //host:port parameters as specified in the DMCONFIG and the `SNACRM` command line parameters in the UBBCONFIG do not agree
2232:ERROR	**Unable to SIGNON to SNACRM %s**	
	DESCRIPTION	An error has occurred while signing on to the CRM.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
2233:ERROR	**Unable to send STACK %s configuration to SNACRM %s**	
	DESCRIPTION	An error has occurred while sending the stack configuration to the CRM.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
2234:ERROR	**Unable to send SNALINK %s configuration to SNACRM %s**	
	DESCRIPTION	An error has occurred while sending the link configuration to the CRM.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.

2235:ERROR	**Unable to send START message to SNACRM %s**	
	DESCRIPTION	An error has occurred while sending the START command to the SNACRM.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
2236:ERROR	**Unable to send SHUTDOWN message to SNACRM %s**	
	DESCRIPTION	An error has occurred while sending the SHUTDOWN command to the SNACRM.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
2237:ERROR	**STATE ERROR occurred during initialization, SHUTTING DOWN**	
	DESCRIPTION	An internal state error has occurred start up.
	ACTION	Contact BEA Customer Support.
2238:ERROR	**Unknown message type received from SNACRM during initialization**	
	DESCRIPTION	An unknown message has been received from SNACRM.
	ACTION	Contact BEA Customer Support.
2239:ERROR	**GATEWAY FAILS to complete connection to the SNACRM during initialization**	
	DESCRIPTION	Connection to SNACRM is incomplete.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
2240:ERROR	**Unable to send SHUTDOWN message to SNACRM %s**	
	DESCRIPTION	eAM gateway is unable to send the SHUTDOWN command to SNACRM.

	ACTION	Examine the ULOG for more information concerning the reason for the failure. The SNACRM process has probably already terminated.
2241:ERROR	**Unable to access codepage table [%s], reason=%s (%d)**	
	DESCRIPTION	eAM gateway is unable to find the file for the listed codepage for the reason specified. If reason is "Error 0", the *(%d)* value is the return code from the codepage read routine. 2 - Can't read from file 3 - Can't allocate memory 4 - Null/invalid pointer 5 - Invalid parameter 10 - Syntax error 11 - Unsupported format version 12 - Unsupported item
	ACTION	Examine the DMCONFIG to determine if the CODEPAGE for a remote domain is specified incorrectly. Correct and retry.
2302:ERROR	**Unable to complete initialization with SNACRM**	
	DESCRIPTION	Connection to SNACRM is incomplete.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
2509:ERROR	**Unable to obtain new convid id for this request**	
	DESCRIPTION	An internal error has been detected, gw_nw_getnxt_cd has returned a failure.
	ACTION	Contact BEA Customer Support.
2561:ERROR	**Action data incorrect - null action index**	
	DESCRIPTION	An internal error has been detected in gw_nw_recv.
	ACTION	Contact BEA Customer Support.

2601:ERROR	**Unable to send data on sna_conv_idx %d**	
	DESCRIPTION	An error has occurred while sending reply data (gw_nw_reply) to the SNACRM.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
2800:ERROR	**Invalid action index**	
	DESCRIPTION	An internal error has been detected in gw_nw_clr_convid.
	ACTION	Contact BEA Customer Support.
2801:ERROR	**Invalid conversation id**	
	DESCRIPTION	An internal error has been detected in gw_nw_clr_convid.
	ACTION	Contact BEA Customer Support.
2802:ERROR	**Invalid action index**	
	DESCRIPTION	An internal error has been detected in gw_nw_getnxt_cd.
	ACTION	Contact BEA Customer Support.
2805:ERROR	**Unable to obtain remote domain info from shared memory**	
	DESCRIPTION	An internal error has been detected, gw_nw_init_env was unable to get needed information from shared memory.
	ACTION	Contact BEA Customer Support.
2806:ERROR	**Unable to obtain snadom info from rdom**	
	DESCRIPTION	An internal error has been detected, gw_nw_init_env was unable to get needed information from shared memory.
	ACTION	Contact BEA Customer Support.
2808:ERROR	**Cannot realloc sna structures**	

	DESCRIPTION	An internal error has been detected, gw_nw_AllocNwCtx was unable to allocate memory needed.
	ACTION	Contact BEA Customer Support.
2816:ERROR	**Memory allocation failure**	
	DESCRIPTION	An internal error has been detected, gw_nw_empty_sndbuf was unable to allocate memory needed.
	ACTION	Contact BEA Customer Support.
2847:ERROR	**SECURITY: Unable to get remote username from shared memory using local principal name %s**	
	DESCRIPTION	A security error has occurred, the user is not authorized to access the remote host.
	ACTION	Contact your local security administrator.
2848:ERROR	**SECURITY: Unable to get local principal name from shared memory using appkey**	
	DESCRIPTION	A security error has occurred, the user is not authorized to access the remote host.
	ACTION	Contact your local security administrator.
2849:ERROR	**SECURITY: Unable to extract password from shared memory**	
	DESCRIPTION	A security error has occurred, the user is not authorized to access the remote host.
	ACTION	Contact your local security administrator.
2850:ERROR	**SECURITY: Unable to get security schedule**	
	DESCRIPTION	A security error has occurred, the user is not authorized to access the remote host.
	ACTION	Contact your local security administrator.
2853:ERROR	**Invalid action index**	

	DESCRIPTION	An internal error has been detected in gw_nw_mk_error.
	ACTION	Contact BEA Customer Support.
2854:ERROR	**Context data incorrect - null context index for action %d**	
	DESCRIPTION	An internal error has been detected in gw_nw_mk_error.
	ACTION	Contact BEA Customer Support.
2860:ERROR	**SECURITY: Unable to get remote user from shared memory**	
	DESCRIPTION	A security error has occurred, the remote user is not authorized to access the local Tuxedo host.
	ACTION	Contact your local security administrator.
2935:ERROR	**Too many unrecoverable errors occurred - deleting action**	
	DESCRIPTION	An error has occurred in gw_nw_mk_error.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
2938:ERROR	**SECURITY: Unable to get appkey for this outbound request**	
	DESCRIPTION	A security error has occurred, the user is not authorized to access the remote host.
	ACTION	Contact your local security administrator.
2939:ERROR	**Deleting the action with the invalid context, %d**	
	DESCRIPTION	An internal error has been detected in gw_nw_snd failure.
	ACTION	Contact BEA Customer Support.
3028:ERROR	**Can’t find transaction node for Network TMS event!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranCommon.
	ACTION	Contact BEA Customer Support.

3029:WARNING	**Shutting down GWSNAX!**	
	DESCRIPTION	A severe error such as a protocol error or memory corruption has occurred and has already been logged. The GWSNAX process will shutdown.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
3030:ERROR	**Unknown input transactional event(%d)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranCvtMsgEvt2TranEvt.
	ACTION	Contact BEA Customer Support.
3031:ERROR	**Invalid input object pointer!**	
	DESCRIPTION	An internal error has been detected in gw_nw_BindStateObject.
	ACTION	Contact BEA Customer Support.
3032:ERROR	**Invalid input object initial state!**	
	DESCRIPTION	An internal error has been detected in gw_nw_BindStateObject.
	ACTION	Contact BEA Customer Support.
3033:ERROR	**Invalid input object type!**	
	DESCRIPTION	An internal error has been detected in gw_nw_BindStateObject.
	ACTION	Contact BEA Customer Support.
3034:ERROR	**Invalid input state object event(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_StateObjectTestEvent.
	ACTION	Contact BEA Customer Support.

3035:ERROR	**Event %s is not allowed for state %s!**	
	DESCRIPTION	An internal error has been detected in gw_nw_StateObjectTestEvent.
	ACTION	Contact BEA Customer Support.
3036:ERROR	**Invalid object event test metric with value %d for %s!**	
	DESCRIPTION	An internal error has been detected in gw_nw_StateObjectTestEvent.
	ACTION	Contact BEA Customer Support.
3037:ERROR	**Invalid input trace object pointer!**	
	DESCRIPTION	An internal error has been detected in gw_nw_InitEventTrace.
	ACTION	Contact BEA Customer Support.
3038:ERROR	**Invalid input trace object pointer!**	
	DESCRIPTION	An internal error has been detected in gw_nw_AddEventTrace.
	ACTION	Contact BEA Customer Support.
3039:ERROR	**Input event %d out of range %d!**	
	DESCRIPTION	An internal error has been detected in gw_nw_AddEventTrace.
	ACTION	Contact BEA Customer Support.
3040:ERROR	**Invalid input trace object pointer!**	
	DESCRIPTION	An internal error has been detected in gw_nw_QueryEventTrace.
	ACTION	Contact BEA Customer Support.
3041:ERROR	**Invalid input trace object pointer!**	
	DESCRIPTION	An internal error has been detected in gw_nw_PrintEventTrace.

	ACTION	Contact BEA Customer Support.
3042:ERROR	**Invalid input event identification(%d)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_EventToString.
	ACTION	Contact BEA Customer Support.
3043:ERROR	**Memory allocation failure**	
	DESCRIPTION	An internal error has been detected in gw_nw_CreateNodeTable.
	ACTION	Contact BEA Customer Support.
3044:WARNING	**Transaction table already exists!**	
	DESCRIPTION	An attempt has been made to create the node table and it already exists.
	ACTION	No action needs to be taken.
3045:ERROR	**Invalid input parameters!**	
	DESCRIPTION	An internal error has been detected in gw_nw_AssociateTranByTXID.
	ACTION	Contact BEA Customer Support.
3046:ERROR	**Requires SNACRM id to create transaction!**	
	DESCRIPTION	An internal error has been detected in gw_nw_AssociateTranByTXID.
	ACTION	Contact BEA Customer Support.
3047:ERROR	**Cannot associate with network transaction!**	
	DESCRIPTION	An internal error has been detected in gw_nw_AssociateTranByTXID.
	ACTION	Contact BEA Customer Support.
3048:ERROR	**Invalid input transaction context!**	

	DESCRIPTION	An internal error has been detected in gw_nw_AssociateTranByTCTXT.
	ACTION	Contact BEA Customer Support.
3049:ERROR	**Invalid input transaction event(%d)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_AssociateTranByTCTXT.
	ACTION	Contact BEA Customer Support.
3050:ERROR	**Invalid input transaction context pointer!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranCreateTCTXT.
	ACTION	Contact BEA Customer Support.
3051:ERROR	**Exceeded local transaction limit(%d)!**	
	DESCRIPTION	Number of concurrent transactions using this domain gateway exceeded the configured limit.
	ACTION	Examine the DMCONFIG file for the MAXTRAN parameter, this number is the limit. If MAXTRAN is not specified, then examine the ubbconfig file for the MAXGTT parameter. In the later case MAXGTT is the limit. Modify the limit if necessary.
3052:ERROR	**Can not create shared memory nettxid!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranCreateTCTXT.
	ACTION	Contact BEA Customer Support.
3053:ERROR	**Create transaction tree node failed!**	
	DESCRIPTION	gw_nw_TranCreateTCTXT failed to create the transaction tree node object. This can be caused by reaching the configured transaction limit or by a memory allocation failure.

	ACTION	Examine the ULOG for more information concerning the reason for the failure. If the cause is the limit being reached, update the configuration. If the cause is a memory allocation failure, update the system and/or the system configuration.
3054:ERROR	**Invalid input transaction handle!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranGetTCTXT.
	ACTION	Contact BEA Customer Support.
3055:ERROR	**Invalid input transaction handle!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranGetTXID.
	ACTION	Contact BEA Customer Support.
3056:ERROR	**Invalid input parameter!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranProcessEvent.
	ACTION	Contact BEA Customer Support.
3057:ERROR	**Unable to calculate transaction handle!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranProcessEvent.
	ACTION	Contact BEA Customer Support.
3058:ERROR	**Input Gateway event not transactional!**	
	DESCRIPTION	The service request inbound from a remote host is transactional but the eAM gateway is not configured as a transactional gateway.
	ACTION	The MAXSYNCLVL specified in the DMCONFIG must be 2 if transactional requests are to be processed.
3059:ERROR	**Transaction does not exist!**	

	DESCRIPTION	An internal error has been detected in gw_nw_TranProcessEvent.
	ACTION	Contact BEA Customer Support.
3060:ERROR	**Invalid input transaction handle!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranProcessEvent.
	ACTION	Contact BEA Customer Support.
3061:ERROR	**Invalid Gateway action identification!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranSetUpExtTmsEvent.
	ACTION	Contact BEA Customer Support.
3062:ERROR	**Decoding input transaction message failed!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranSetUpExtTmsEvent.
	ACTION	Contact BEA Customer Support.
3063:ERROR	**SNACRM verb %d is not for TMS event!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranSetUpExtTmsEvent.
	ACTION	Contact BEA Customer Support.
3064:ERROR	**Invalid input transaction context!!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranRelease.
	ACTION	Contact BEA Customer Support.
3065:ERROR	**Transaction for context(%d) not found!**	
	DESCRIPTION	The specified context for the transaction to be released could not be found by gw_nw_TranRelease.

	ACTION	Examine the ULOG for more information concerning the reason for the failure.
3066:ERROR	**Cannot disassociate with network transaction!**	
	DESCRIPTION	An error has been returned by gw_tx_end.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
3067:ERROR	**Cannot set remote domain for transaction!**	
	DESCRIPTION	An error has been returned by gw_tx_set_rdom. Several possibilities exist for this failure. 1. The process is unable to obtain the shared memory lock. 2. The configured limit for the number of remote domains that can be involved in the same transaction has been reached. This limit is specified in the MAXRDTRAN in the DMCONFIG.
	ACTION	Examine the DMCONFIG and compare with the number of remote domains which might become involved in the same transaction. Update the DMCONFIG if necessary. Also examine the ULOG for more information concerning possible reasons for the failure.
3068:ERROR	**Invalid input transaction handle!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranGetNodePtr.
	ACTION	Contact BEA Customer Support.
3069:ERROR	**Transaction handle not in use!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranGetNodePtr.
	ACTION	Contact BEA Customer Support.
3070:ERROR	**Transaction processing not yet initialize!**	

	DESCRIPTION	An internal error has been detected in gw_nw_TranGetNodePtr.
	ACTION	Contact BEA Customer Support.
3071:ERROR	**Defined transaction limit %d reached!**	
	DESCRIPTION	The number of concurrent transactions using this domain gateway has exceeded the limit.
	ACTION	Examine the DMCONFIG file for the MAXTRAN parameter, this number is the limit. If MAXTRAN is not specified, then examine the ubbconfig file for the MAXGTT parameter. In the later case MAXGTT is the limit. Modify the limit if necessary.
3072:ERROR	**Allocate SNACRM branch structure failed!**	
	DESCRIPTION	An internal error has been detected in gw_nw__gw_nw_AllocateNode.
	ACTION	Contact BEA Customer Support.
3073:ERROR	**Unable to bind outbound node state object!**	
	DESCRIPTION	A failure was returned by gw_nw_BindStateObject.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
3074:ERROR	**Unable to bind inbound node state object!**	
	DESCRIPTION	A failure was returned by gw_nw_BindStateObject.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
3075:ERROR	**Unable to bind recover node state object!**	
	DESCRIPTION	A failure was returned by gw_nw_BindStateObject.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.

3076:ERROR	**Invalid event %s to create transaction tree node!**	
	DESCRIPTION	An internal error has been detected in _gw_nw_AllocateNode.
	ACTION	Contact BEA Customer Support.
3077:ERROR	**Memory allocation failure!**	
	DESCRIPTION	An internal error has been detected in gw_nw_AllocBranch.
	ACTION	Contact BEA Customer Support.
3078:ERROR	**Event(%s) is not allowed to create SNACRM branch!**	
	DESCRIPTION	An internal error has been detected in gw_nw_AllocBranch.
	ACTION	Contact BEA Customer Support.
3079:ERROR	**Free more branch than transaction tree node has!**	
	DESCRIPTION	An internal error has been detected in gw_nw_FreeBranch.
	ACTION	Contact BEA Customer Support.
3080:ERROR	**Protocol violation, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_3.
	ACTION	Contact BEA Customer Support.
3081:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_4.
	ACTION	Contact BEA Customer Support.

3082:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_5.
	ACTION	Contact BEA Customer Support.
3083:ERROR	**Branch in a bad state(%s) to issue prepare!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_5.
	ACTION	Contact BEA Customer Support.
3084:WARNING	**Inconsistent branch state for prepare(%d, %d)!**	
	DESCRIPTION	The transaction branch is in an inconsistent state for issuing a prepare. It will be assumed that they are all prepared.
	ACTION	No action needs to be taken.
3085:INFO	**Transaction node is deallocated because of ABEND!**	
	DESCRIPTION	Informational message. eAM gateway detected an ABEND condition or received an ABEND event from the SNACRM which caused the node object to be destroyed.
	ACTION	No action needs to be taken.
3086:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_7.
	ACTION	Contact BEA Customer Support.
3087:ERROR	**Branch in a bad state(%s) to issue prepare!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_7.
	ACTION	Contact BEA Customer Support.

3088:WARNING	**Inconsistent branch state for rollback(%d, %d)!**	
	DESCRIPTION	The transaction branch was in an inconsistent state for issuing a rollback. It will be assumed that they are all rolled back.
	ACTION	No action needs to be taken.
3089:WARNING	**Inconsistent branch accounting for rollback(%d, %d, %d)!**	
	DESCRIPTION	The transaction branch accounting was inconsistent while issuing a rollback.
	ACTION	No action needs to be taken.
3090:WARNING	**Incorrect branch accounting for rollback(%d)!**	
	DESCRIPTION	The transaction branch accounting was incorrect while issuing a rollback.
	ACTION	No action needs to be taken.
3091:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_8.
	ACTION	Contact BEA Customer Support.
3092:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_9.
	ACTION	Contact BEA Customer Support.
3093:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_10.
	ACTION	Contact BEA Customer Support.

3094:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_11.
	ACTION	Contact BEA Customer Support.
3095:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_12.
	ACTION	Contact BEA Customer Support.
3096:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_13.
	ACTION	Contact BEA Customer Support.
3097:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_14.
	ACTION	Contact BEA Customer Support.
3098:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_15.
	ACTION	Contact BEA Customer Support.
3099:ERROR	**Retired node method!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_Retired.
	ACTION	Contact BEA Customer Support.

3100:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_16.
	ACTION	Contact BEA Customer Support.
3101:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_17.
	ACTION	Contact BEA Customer Support.
3102:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_18.
	ACTION	Contact BEA Customer Support.
3103:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_19.
	ACTION	Contact BEA Customer Support.
3104:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_20.
	ACTION	Contact BEA Customer Support.
3105:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_21.
	ACTION	Contact BEA Customer Support.

3106:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_22.
	ACTION	Contact BEA Customer Support.
3107:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_23.
	ACTION	Contact BEA Customer Support.
3108:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_24.
	ACTION	Contact BEA Customer Support.
3109:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_25.
	ACTION	Contact BEA Customer Support.
3110:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_26.
	ACTION	Contact BEA Customer Support.
3111:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_27.
	ACTION	Contact BEA Customer Support.

3112:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_28.
	ACTION	Contact BEA Customer Support.
3113:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_29.
	ACTION	Contact BEA Customer Support.
3114:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_30.
	ACTION	Contact BEA Customer Support.
3115:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_31.
	ACTION	Contact BEA Customer Support.
3116:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_32.
	ACTION	Contact BEA Customer Support.
3117:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_33.
	ACTION	Contact BEA Customer Support.

3118:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_39.
	ACTION	Contact BEA Customer Support.
3119:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_40.
	ACTION	Contact BEA Customer Support.
3120:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_42.
	ACTION	Contact BEA Customer Support.
3121:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_43.
	ACTION	Contact BEA Customer Support.
3122:ERROR	**Encoding transaction message for verb(%d) failed!**	
	DESCRIPTION	A failure was returned by snacrm_tx_msg.
	ACTION	Contact BEA Customer Support.
3123:ERROR	**SNACRM(%d) in bad state(%s) to issue verb(%d)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeSendToBranch.
	ACTION	Contact BEA Customer Support.

3124:ERROR	**Send verb %d to snacrm failed!**	
	DESCRIPTION	A failure has been detected while sending the specified verb to the SNACRM.
	ACTION	Examine the ULOG for more information concerning the reason for the failure. Determine if the SNACRM process is still active.
3125:ERROR	**Encoding transactional message for verb(%d) failed!**	
	DESCRIPTION	A failure was returned by snacrm_tx_msg.
	ACTION	Contact BEA Customer Support.
3126:ERROR	**SNACRM(%d) in bad state(%s) to issue verb(%d)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeSendToSubordinate.
	ACTION	Contact BEA Customer Support.
3127:ERROR	**Send verb(%d) to SNACRM failed!**	
	DESCRIPTION	A failure has been detected while sending the specified verb to the SNACRM.
	ACTION	Examine the ULOG for more information concerning the reason for the failure. Determine if the SNACRM process is still active.
3128:ERROR	**Encoding transactional message for verb(%d) failed!**	
	DESCRIPTION	A failure was returned by snacrm_tx_msg.
	ACTION	Contact BEA Customer Support.
3129:ERROR	**SNACRM(%d) in bad state(%s) to issue verb(%d)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeSendToSubordinate.
	ACTION	Contact BEA Customer Support.

3130:ERROR	**Send verb(%d) to SNACRM failed!**	
	DESCRIPTION	A failure has been detected while sending the specified verb to the SNACRM.
	ACTION	Examine the ULOG for more information concerning the reason for the failure. Determine if the SNACRM process is still active.
3131:ERROR	**Protocol violation, branch id[%d] at state(%s) receive event(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_BranchMethod_1.
	ACTION	Contact BEA Customer Support.
3132:ERROR	**Memory allocation error!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranRestart.
	ACTION	Contact BEA Customer Support.
3133:ERROR	**Unable to get shared memory semaphore lock!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranRestart.
	ACTION	Contact BEA Customer Support.
3134:ERROR	**Unable to find transaction on GTT!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranRestart.
	ACTION	Contact BEA Customer Support.
3135:ERROR	**SNAX group is not participating the transaction!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranRestart.
	ACTION	Contact BEA Customer Support.

3136:ERROR	**Unable to setup transaction recovery!**	
	DESCRIPTION	A failure was returned by gw_nw_TranRecover.
	ACTION	Contact BEA Customer Support.
3137:ERROR	**Unable to setup transaction recovery!**	
	DESCRIPTION	A failure was returned by gw_nw_TranRecover.
	ACTION	Contact BEA Customer Support.
3138:ERROR	**Unable to create recovery action!**	
	DESCRIPTION	A failure was returned by gw_new_action.
	ACTION	Contact BEA Customer Support.
3139:ERROR	**Unable to create recovery action!**	
	DESCRIPTION	A failure was returned by gw_chld_action.
	ACTION	Contact BEA Customer Support.
3140:ERROR	**Unable to setup transaction recovery!**	
	DESCRIPTION	A failure was returned by gw_nw_TranRecover.
	ACTION	Contact BEA Customer Support.
3141:ERROR	**Unable to create recovery action!**	
	DESCRIPTION	A failure was returned by gw_new_action.
	ACTION	Contact BEA Customer Support.
3142:ERROR	**Unable to create recovery action!**	
	DESCRIPTION	A failure was returned by gw_chld_action.
	ACTION	Contact BEA Customer Support.
3143:ERROR	**Unable to setup transaction recovery!**	
	DESCRIPTION	A failure was returned by gw_nw_TranRecover.
	ACTION	Contact BEA Customer Support.

3144:ERROR	**Unable to setup transaction recovery!**	
	DESCRIPTION	A failure was returned by gw_nw_TranRecover.
	ACTION	Contact BEA Customer Support.
3145:ERROR	**Unable to release shared memory lock!**	
	DESCRIPTION	A failure was returned by _gw_shmunlock.
	ACTION	Contact BEA Customer Support.
3146:ERROR	**Unable to find SNACRM for superior remote domain(%s)!**	
	DESCRIPTION	Failed to find the remote domain definition in the configuration in order to get the SNACRM identification.
	ACTION	Examine the DMCONFIG for changes between the boots that would modify or remove the remote definition. If no such change has occurred, contact BEA Customer Support.
3147:ERROR	**Could not find Remote Domain(%s) to do recovery!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranRecover.
	ACTION	Contact BEA Customer Support.
3148:ERROR	**Could not find Remote Domain to do recovery!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranRecover.
	ACTION	Contact BEA Customer Support.

3149:ERROR	**Could not create transaction tree structure!**	
	DESCRIPTION	Unable to allocate a transaction tree node structure to represent the transaction recovery object.
	ACTION	Examine the configuration to determine if there has been a change of MAXTRAN in DMCONFIG and/or a change of MAXGTT in UBBCONFIG. This condition may occur if there is a decrease in the number of transactions allowed between boots while many transactions must be recovered. If no changes have been made, contact BEA Customer Support.
3150:ERROR	**Unable to create SNACRM branch structure!**	
	DESCRIPTION	A failure was returned by _gw_nw_AllocBranch.
	ACTION	Contact BEA Customer Support.
3151:ERROR	**Unable to create SNACRM branch structure!**	
	DESCRIPTION	A failure was returned by _gw_nw_AllocBranch.
	ACTION	Contact BEA Customer Support.
3152:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_44.
	ACTION	Contact BEA Customer Support.
3153:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_45.
	ACTION	Contact BEA Customer Support.
3154:ERROR	**Invalid Gateway action, receive event(%s) at state(%s)!**	
	DESCRIPTION	An internal error has been detected in gw_nw_NodeMethod_46.
	ACTION	Contact BEA Customer Support.

3155:ERROR	**Invalid transaction context(%d) from SNACRM!**	
	DESCRIPTION	An internal error has been detected in gw_nw_TranSetUpExtTmsEvent.
	ACTION	Contact BEA Customer Support.
3156:WARNING	**Undefined instrumentation register %d specification, ignored!**	
	DESCRIPTION	An internal error has been detected in gw_nw_SetUpInstrumentation.
	ACTION	Contact BEA Customer Support.
3157:INFO	**CONNECTED TO SNACRM**	
	DESCRIPTION	The eAM gateway process has successfully started the gateway connection to the SNACRM process.
	ACTION	No action needs to be taken.
3158:WARNING	**SNACRM Link %s Inoperable**	
	DESCRIPTION	The specified link did not start up correctly.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
3159:ERROR	**INVALID STARTTYPE (%s) - calling shutdown**	
	DESCRIPTION	The STARTTYPE specified in the DMCONFIG must be AUTO or COLD.
	ACTION	Correct the value specified in the DMCONFIG and restart.
3500:ERROR	**Unable to find remote service definition link**	
	DESCRIPTION	An internal error has been detected in gw_nw_decode.
	ACTION	Contact BEA Customer Support.

3501:ERROR	**Unable to find local service definition link**	
	DESCRIPTION	An internal error has been detected in gw_nw_decode.
	ACTION	Contact BEA Customer Support.
3502:ERROR	**Buffer type (%s,%s) not defined in the buffer type switch!**	
	DESCRIPTION	The buffer type associated with the inbound message is not specified in the type switch.
	ACTION	Examine the application and the configuration to determine which buffer type is being used. eAM gateway only supports standard Tuxedo buffer types over the domain gateway. If the buffer type is VIEW, VIEW32, then also examine the environment variables VIEWDIR, VIEWFILES, VIEW32DIR, or VIEW32FILES.
3503:ERROR	**Failed to tpalloc() STRING buffer of size %ld**	
	DESCRIPTION	No buffer of type STRING was returned by tpalloc for the size specified.
	ACTION	If the size specified is 1, then a length of zero was received. Check the application to determine if it has an error, otherwise contact BEA Customer Support.
3504:ERROR	**Failed to tpalloc() CARRAY/X_OCTET buffer of size %ld**	
	DESCRIPTION	No buffer of type CARRAY or X_OCTET was returned by tpalloc for the size specified.
	ACTION	If the size specified is 1, then a length of zero was received. Check the application to determine if it has an error, otherwise contact BEA Customer Support.

3505:ERROR	**Input buffer too big(%ld)!**	
	DESCRIPTION	The buffer received of the size specified, exceeded the maximum allowed of 32767.
	ACTION	Check the application to determine if it has an error, otherwise contact BEA Customer Support.
3506:ERROR	**Missing subtype specification for %s**	
	DESCRIPTION	The buffer type specified of VIEW, VIEW32, X_C_TYPE, or X_COMMON requires that a subtype be specified.
	ACTION	Examine the application and the DMCONFIG and correct the problem.
3507:ERROR	**Malloc temporary buffer of size(%ld) failed!**	
	DESCRIPTION	An internal error has been detected in **gw_nw_decode**.
	ACTION	Contact BEA Customer Support.
3508:ERROR	**Convert buffer type(%s,%s) failed, convert failure!**	
	DESCRIPTION	The conversion of the buffer received from the remote host was unsuccessful.
	ACTION	Examine the application to determine if it has an error, examine the DMCONFIG definition for any errors in the buffer type specification. If no errors are found, contact BEA Customer Support.
3509:ERROR	**Convert buffer type(%s,%s) failed, length too big!**	
	DESCRIPTION	The conversion of the buffer received from the remote host was unsuccessful due to the resulting size exceeding the maximum of 32767.
	ACTION	Examine the application to determine if it has an error, examine the DMCONFIG definition for any errors in the buffer type specification. If no errors are found, contact BEA Customer Support.

3510:ERROR	**Failed to tpalloc() buffer (%s,%s) of size %ld**	
	DESCRIPTION	No buffer was returned by tpalloc for the size specified to hold the converted buffer data.
	ACTION	Examine the application to determine if it has an error, otherwise contact BEA Customer Support.
3511:ERROR	**Missing subtype for FML type buffer!**	
	DESCRIPTION	The buffer type specified of FML requires that a subtype be specified.
	ACTION	Examine the application and the configuration to determine which buffer type is being used. eAM gateway only supports standard Tuxedo buffer types over the domain gateway. If the buffer type is VIEW, VIEW32, then also examine the environment variables VIEWDIR, VIEWFILES, VIEW32DIR, or VIEW32FILES.
3512:ERROR	**Malloc temporary buffer of size(%ld) failed!**	
	DESCRIPTION	An internal error has been detected in gw_nw_decode.
	ACTION	Contact BEA Customer Support.
3513:ERROR	**Malloc failed, unable to convert!**	
	DESCRIPTION	The conversion of the buffer received from the remote host was unsuccessful.
	ACTION	Examine the application to determine if it has an error, examine the DMCONFIG definition for any errors in the buffer type specification. If no errors are found, contact BEA Customer Support.
3514:ERROR	**Failed to malloc temporary buffer of size %ld**	
	DESCRIPTION	An internal error has been detected in gw_nw_decode.
	ACTION	Contact BEA Customer Support.

3515:ERROR	**Convert buffer type(%s,%s) failed, convert failure!**	
	DESCRIPTION	The conversion of the buffer received from the remote host was unsuccessful.
	ACTION	Examine the application to determine if it has an error, examine the DMCONFIG definition for any mistakes in the buffer type specification. If no errors are found, contact BEA Customer Support.
3516:ERROR	**Failed to tpalloc() buffer (%s,%s) of size %ld**	
	DESCRIPTION	No buffer was returned by tpalloc for the size specified to hold the converted buffer data.
	ACTION	Check the application to determine if it has an error, otherwise contact BEA Customer Support.
3517:ERROR	**Missing subtype definition for FML32 buffer type!**	
	DESCRIPTION	The buffer type specified of FML32 requires that a subtype be specified.
	ACTION	Correct the DMCONFIG definition.
3518:ERROR	**Malloc temporary of size(%ld) failed!**	
	DESCRIPTION	An internal error has been detected in gw_nw_decode.
	ACTION	Contact BEA Customer Support.
3519:ERROR	**Convert FML32 buffer type(%s,%s) failed, convert failure!**	
	DESCRIPTION	The conversion of the buffer received from the remote host was unsuccessful.
	ACTION	Examine the application to determine if it has an error, examine the DMCONFIG definition for any errors in the buffer type specification. If no errors are found, contact BEA Customer Support.

3520:ERROR	**Malloc temporary buffer of size(%ld) failed!**	
	DESCRIPTION	An internal error has been detected in gw_nw_decode.
	ACTION	Contact BEA Customer Support.
3521:ERROR	**Convert FML32 buffer with subtype(%s) failed!**	
	DESCRIPTION	The conversion of the buffer received from the remote host was unsuccessful.
	ACTION	Examine the application to determine if it has an error, examine the DMCONFIG definition for any errors in the buffer type specification. If no errors are found, contact BEA Customer Support.
3522:ERROR	**Failed to tpalloc() FML32 buffer of size %d for %s!**	
	DESCRIPTION	An internal error has been detected in gw_nw_decode.
	ACTION	Contact BEA Customer Support.
3523:ERROR	**Fail to convert for FML buffer(%s), exceeding limit!**	
	DESCRIPTION	Failed to convert the FML buffer; the buffer size required to process the conversion exceeds the limit.
	ACTION	Examine the application and the view for this buffer for any errors. If no errors are found, contact BEA Customer Support.
3524:ERROR	**Fail to convert for FML32 buffer(%s), exceeding limit!**	
	DESCRIPTION	Failed to convert the FML32 buffer; the buffer size required to process the conversion exceeds the limit.
	ACTION	Examine the application and the view for this buffer for any errors. If no errors are found, contact BEA Customer Support.

3525:ERROR	**Fail to retrieve remote service definition!**	
	DESCRIPTION	Unable to find the remote service definition in shared memory.
	ACTION	Examine the DMCONFIG remote service definitions. Correct any errors found. If no error is found, contact BEA Customer Support.
3526:ERROR	**Fail to retrieve local service definition!**	
	DESCRIPTION	Unable to find the local service definition in shared memory.
	ACTION	Examine the DMCONFIG local service definitions. Correct any errors found. If no error is found, contact BEA Customer Support.
3527:ERROR	**Must specify subtype for %s**	
	DESCRIPTION	The buffer type specified of VIEW, VIEW32, X_C_TYPE, or X_COMMON requires that a subtype be specified.
	ACTION	Correct any errors in the DMCONFIG buffer type definitions. If no errors are found, contact BEA Customer Support.
3528:ERROR	**Malloc temporary buffer of size(%d) failed!**	
	DESCRIPTION	An internal error has been detected in gw_nw_encode.
	ACTION	Contact BEA Customer Support.
3529:ERROR	**Convert buffer type %s with subtype %s failed(rc=%d)!**	
	DESCRIPTION	The conversion of the buffer received from Tuxedo was unsuccessful.
	ACTION	Examine the application to determine if it has an error, examine the DMCONFIG definition for any errors in the buffer type specification. If no errors are found, contact BEA Customer Support.

3530:ERROR	**Convert buffer type %s with subtype %s failed(rc=%d)!**	
	DESCRIPTION	The conversion of the buffer received from Tuxedo was unsuccessful due to the resulting size exceeding the maximum of 32767.
	ACTION	Examine the application to determine if it has an error, examine the DMCONFIG definition for any errors in the buffer type specification. If no errors are found, contact BEA Customer Support.
3531:ERROR	**Malloc temporary buffer of size(%d) failed!**	
	DESCRIPTION	An internal error has been detected in gw_nw_encode.
	ACTION	Contact BEA Customer Support.
3532:ERROR	**FML buffer type requires subtype to be specified!**	
	DESCRIPTION	The buffer type specified of FML requires that a subtype be specified.
	ACTION	Correct the DMCONFIG definition.
3533:ERROR	**Malloc temporary buffer of size(%d) failed!**	
	DESCRIPTION	An internal error has been detected in gw_nw_encode.
	ACTION	Contact BEA Customer Support.
3534:ERROR	**Convert FML buffer with subtype %s failed(Ferror=%d)!**	
	DESCRIPTION	The conversion of the FML buffer received from Tuxedo was unsuccessful. Ferror is the return code from the FML conversion routine.
	ACTION	Examine the application to determine if it has an error, examine the DMCONFIG definition for any errors in the buffer type specification. If no errors are found, contact BEA Customer Support.

3535:ERROR	**Malloc temporary buffer of size(%d) failed!**	
	DESCRIPTION	An internal error has been detected in gw_nw_encode.
	ACTION	Contact BEA Customer Support.
3536:ERROR	**Convert FML buffer with subtype %s failed(Ferror=%d)!**	
	DESCRIPTION	The conversion of the FML buffer received from Tuxedo was unsuccessful. Ferror is the return code from the FML conversion routine.
	ACTION	Examine the application to determine if it has an error, examine the DMCONFIG definition for any errors in the buffer type specification. If no errors are found, contact BEA Customer Support.
3537:ERROR	**Malloc buffer of size(%d) failed!**	
	DESCRIPTION	An internal error has been detected in gw_nw_encode.
	ACTION	Contact BEA Customer Support.
3538:ERROR	**FML32 buffer type requires subtype to be specified!**	
	DESCRIPTION	The buffer type specified of FML32 requires that a subtype be specified.
	ACTION	Correct the DMCONFIG definition.
3539:ERROR	**Malloc temporary buffer of size(%d) failed!**	
	DESCRIPTION	An internal error has been detected, gw_nw_Alloc was unable to allocate memory needed.
	ACTION	Contact BEA Customer Support.

3540:ERROR	**Convert FML32 buffer with subtype %s failed(Ferror=%d)!**	
	DESCRIPTION	The conversion of the FML32 buffer received from Tuxedo was unsuccessful. Ferror is the return code from the FML conversion routine.
	ACTION	Examine the application to determine if it has an error, examine the DMCONFIG definition for any errors in the buffer type specification. If no errors are found, contact BEA Customer Support.
3541:ERROR	**Malloc temporary buffer of size(%d) failed!**	
	DESCRIPTION	An internal error has been detected, gw_nw_Alloc was unable to allocate memory needed.
	ACTION	Contact BEA Customer Support.
3542:ERROR	**Convert FML32 buffer with subtype %s failed(Ferror=%d)!**	
	DESCRIPTION	The conversion of the FML32 buffer received from Tuxedo was unsuccessful. Ferror is the return code from the FML conversion routine.
	ACTION	Examine the application to determine if it has an error, examine the DMCONFIG definition for any errors in the buffer type specification. If no errors are found, contact BEA Customer Support.
4000:ERROR	**GP_SEND flow controlled for GPND %d**	
	DESCRIPTION	An internal error has been detected, gw_nw_proto_flow has detected a flow control callback from gp_send.
	ACTION	Contact BEA Customer Support.
4100:ERROR	**Maximum number of pointers exceeded**	
	DESCRIPTION	An internal error has been detected. The maximum number of pointers has exceeded 500.
	ACTION	Contact BEA Customer Support.

4124:ERROR	**Unable to format snacrm_acall**	
	DESCRIPTION	An internal error has been detected, snacrm_acall has returned an error.
	ACTION	Contact BEA Customer Support.
4125:ERROR	**Unable to send data on sna_conv_idx %d, gpnd = %d**	
	DESCRIPTION	An error has occurred while sending acall data (gw_nw_acall) to the SNACRM.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
4126:ERROR	**Unable to format snacrm_connect**	
	DESCRIPTION	An internal error has been detected, snacrm_connect has returned an error.
	ACTION	Contact BEA Customer Support.
4127:ERROR	**Unable to send data on sna_conv_idx %d, gpnd = %d**	
	DESCRIPTION	An error has occurred while sending connect data (gw_nw_connect) to the SNACRM.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
4128:ERROR	**Unable to format snacrm_send_data**	
	DESCRIPTION	An internal error has been detected, snacrm_send has returned an error.
	ACTION	Contact BEA Customer Support.
4130:ERROR	**Unable to format snacrm_connect**	
	DESCRIPTION	An internal error has been detected, snacrm_connect_rsp has returned an error.
	ACTION	Contact BEA Customer Support.

5019:ERROR	**Send event for RRCORR(%ld) failed!**	
	DESCRIPTION	An internal error has been detected, gw_nw_SendEvent has returned an error.
	ACTION	Contact BEA Customer Support.
5020:ERROR	**Encode ACALL_RSP failed, RRCORR(%ld)!**	
	DESCRIPTION	An internal error has been detected, snacrm_acall_rsp has returned an error.
	ACTION	Contact BEA Customer Support.
5021:ERROR	**Encode reply failed!**	
	DESCRIPTION	An error was detected while attempting to encode the reply message data destined for the remote host.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
5022:ERROR	**Invalid network descriptor(%d)!**	
	DESCRIPTION	The network descriptor this context is invalid.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.

5023:ERROR	**Could not get transaction**	
	DESCRIPTION	Failed to associate a transactional ACALL service request to a transaction object. The most likely reasons for this failure are: 1. The locally configured limit has been exceeded, or 2. Memory allocation has failed to create the transaction object.
	ACTION	Determine if the application is licensed to do transactions. Examine the DMCONFIG file for the MAXTRAN parameter, this number is the limit. If MAXTRAN is not specified, then examine the ubbconfig file for the MAXGTT parameter. In the later case MAXGTT is the limit. Modify the limit if necessary. Verify that the remote domain MAXSYNCLVL is set to 2.
5024:ERROR	**Could not get API**	
	DESCRIPTION	An internal error has been detected. The API information is invalid.
	ACTION	Contact BEA Customer Support.
5025:ERROR	**Could not get transaction**	
	DESCRIPTION	Failed to associate a transactional CONNECT service request to a transaction object. The most likely reasons for this failure are: 1. The locally configured limit has been exceeded, or 2. Memory allocation has failed to create the transaction object
	ACTION	Determine if the application is licensed to do transactions. Examine the DMCONFIG file for the MAXTRAN parameter, this number is the limit. If MAXTRAN is not specified, then examine the ubbconfig file for the MAXGTT parameter. In the later case MAXGTT is the limit. Modify the limit if necessary. Verify that the remote domain MAXSYNCLVL is set to 2.

5026:ERROR	**Could not get API**	
	DESCRIPTION	An internal error has been detected. The API information is invalid.
	ACTION	Contact BEA Customer Support.
5027:ERROR	**Could not get API**	
	DESCRIPTION	An internal error has been detected. The API information is invalid.
	ACTION	Contact BEA Customer Support.
5028:ERROR	**Protocol violation, shutdown gateway!**	
	DESCRIPTION	An internal error has been detected by gw_nw_crmrcv.
	ACTION	Contact BEA Customer Support.
5029:ERROR	**Link status down for remote domain %s**	
	DESCRIPTION	The link status returned by the SNACRM is LINK_DOWN.
	ACTION	Determine if the stack is active and check that the session between the stack provider and the remote host is active.
5030:ERROR	**Link status pending for remote domain %s**	
	DESCRIPTION	The link status returned by the SNACRM is LINK_PENDING. Transaction recovery for the link is still in progress following a restart. Only non-transactional requests may be sent to this link.
	ACTION	Wait until recovery is complete before requesting transactional services on this link.

5031:ERROR	**Transaction not allowed, request rejected!**	
	DESCRIPTION	Failed to send a transactional ACALL request to the remote domain because MAXSYNCLVL is not 2.
	ACTION	Determine the following things: 1. eAM gateway is licensed for transactions, and 2. DMCONFIG specifies MAXSYNCLVL=2 for the link.
5032:ERROR	**Transaction not allowed, request rejected!**	
	DESCRIPTION	Failed to send a transactional CONNECT request to the remote domain because MAXSYNCLVL is not 2.
	ACTION	Determine the following things: 1. eAM gateway is licensed for transactions, and 2. DMCONFIG specifies MAXSYNCLVL=2 for the link.
5033:ERROR	**Protocol violation, shutdown gateway!**	
	DESCRIPTION	An internal error has been detected by gw_nw_crmrcv.
	ACTION	Contact BEA Customer Support.
5034:ERROR	**Allocate context failed!**	
	DESCRIPTION	An internal error has been detected. An error was returned by gw_nw_AllocNwCtx.
	ACTION	Contact BEA Customer Support.
5035:ERROR	**Enter shutdown**	
	DESCRIPTION	An internal error has been detected by gw_nw_crmrcv.
	ACTION	Contact BEA Customer Support.

5036:ERROR	**Invalid conversation identification, shutdown gateway!**	
	DESCRIPTION	An internal error has been detected by gw_nw_convsend.
	ACTION	Contact BEA Customer Support.
5037:ERROR	**Unknown message type, shutdown gateway!**	
	DESCRIPTION	An internal error has been detected by gw_nw_crmrcv.
	ACTION	Contact BEA Customer Support.
5038:ERROR	**LINK_STATUS state error, shutdown gateway!**	
	DESCRIPTION	An internal error has been detected by gw_nw_crmrcv.
	ACTION	Contact BEA Customer Support.
5039:ERROR	**Protocol violation, shutdown gateway!**	
	DESCRIPTION	An internal error has been detected by gw_nw_crmrcv.
	ACTION	Contact BEA Customer Support.
5040:ERROR	**Enter shutdown**	
	DESCRIPTION	An internal error has been detected, an error was returned by gw_nw_getsncrm_idx.
	ACTION	Contact BEA Customer Support.
5041:ERROR	**Trans Event Error shutdown**	
	DESCRIPTION	An internal error has been detected, an error was returned by gw_nw_TranSetUpExtTmsEvent.
	ACTION	Contact BEA Customer Support.

5042:WARNING	**SIGNON identify failed, shutdown gateway!**	
	DESCRIPTION	SNACRM returned an error to the SIGNON request by eAM gateway.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
5043:ERROR	**Trans Event Error**	
	DESCRIPTION	An internal error has been detected. An error was returned by gw_nw_AssociateTranBy.
	ACTION	Contact BEA Customer Support.
5044:ERROR	**SIGNON state error, shutdown gateway!**	
	DESCRIPTION	An internal error has been detected, an error was returned by _gw_nw_snacrm_nextop.
	ACTION	Contact BEA Customer Support.
5045:ERROR	**STARTUP state error, shutdown gateway!**	
	DESCRIPTION	An internal error has been detected, an error was returned by _gw_nw_snacrm_nextop.
	ACTION	Contact BEA Customer Support.
5046:WARNING	**LINK_STATUS decode error, shutdown gateway!**	
	DESCRIPTION	An internal error has been detected. An error was returned by snacrm_linkstatus_rcv.
	ACTION	Contact BEA Customer Support.
5047:WARNING	**LINK_STATUS status error, shutdown gateway!**	
	DESCRIPTION	An internal error has been detected by gw_nw_crmrcv.
	ACTION	Contact BEA Customer Support.

5048:WARNING	**LINK_CONFIG state error, shutdown gateway!**	
	DESCRIPTION	An internal error has been detected, an error was returned by _gw_nw_snacrm_nextop.
	ACTION	Contact BEA Customer Support.
5049:WARNING	**STACK_CONFIG state error, shutdown gateway!**	
	DESCRIPTION	An internal error has been detected, an error was returned by _gw_nw_snacrm_nextop.
	ACTION	Contact BEA Customer Support.
5050:WARNING	**SHUTDOWN message received, shutdown gateway!**	
	DESCRIPTION	eAM gateway has received a SHUTDOWN request from the SNACRM and will shut down immediately.
	ACTION	Examine the ULOG for more information concerning the reason for the shutdown.
5051:ERROR	**LOG_DATA decode error, shutdown gateway!**	
	DESCRIPTION	An internal error has been detected. An error was returned by snacrm_log_data_rcv.
	ACTION	Contact BEA Customer Support.
5052:ERROR	**LOG_DATA state error(%d), shutdown gateway!**	
	DESCRIPTION	An internal error has been detected by gw_nw_crmrcv.
	ACTION	Contact BEA Customer Support.
5053:ERROR	**Protocol violation, shutdown gateway!**	
	DESCRIPTION	An internal error has been detected by gw_nw_crmrcv. Inbound ACALL request contained SEND_INVITE for function DPL.
	ACTION	Contact BEA Customer Support.

5054:ERROR	**Invalid action index**	
	DESCRIPTION	An internal error has been detected by gw_nw_InBoundConnectRqst.
	ACTION	Contact BEA Customer Support.
5055:ERROR	**Invalid action index**	
	DESCRIPTION	An internal error has been detected by gw_nw_OutBoundConnectRsp.
	ACTION	Contact BEA Customer Support.
5056:ERROR	**Protocol violation, shutdown gateway!**	
	DESCRIPTION	An internal error has been detected by gw_nw_crmrcv.
	ACTION	Contact BEA Customer Support.
5057:ERROR	**SEND_DATA decode error, shutdown gateway!**	
	DESCRIPTION	An internal error has been detected by gw_nw_crmrcv.
	ACTION	Contact BEA Customer Support.
5058:ERROR	**Cannot create action to send remote disconnect**	
	DESCRIPTION	An internal error has been detected, an error was returned by gw_new_action.
	ACTION	Contact BEA Customer Support.
5059:ERROR	**ACALL decode error, shutdown gateway!**	
	DESCRIPTION	An internal error has been detected by gw_nw_crmrcv.
	ACTION	Contact BEA Customer Support.

5060:ERROR	**Invalid action index**	
	DESCRIPTION	An internal error has been detected by gw_nw_InBoundAcallRqst.
	ACTION	Contact BEA Customer Support.
5061:ERROR	**Invalid action index**	
	DESCRIPTION	An internal error has been detected by gw_nw_OutBoundAcallRsp.
	ACTION	Contact BEA Customer Support.
5062:ERROR	**Failed to find local service(%s) definition!**	
	DESCRIPTION	Unable to find the local service definition for the inbound request.
	ACTION	Examine the DMCONFIG local service definitions. Correct any errors found. If no error is found, contact BEA Customer Support.
5063:ERROR	**Invalid link index**	
	DESCRIPTION	An internal error has been detected, an error was returned by _gw_nw_set_link_idx.
	ACTION	Contact BEA Customer Support.
5064:ERROR	**Invalid query_rsp index**	
	DESCRIPTION	An internal error has been detected by gw_nw_QueryProcess.
	ACTION	Contact BEA Customer Support.
5065:ERROR	**Enter shutdown**	
	DESCRIPTION	A failure has been detected while sending the SNACRM_QSVC_RSP verb to the SNACRM.
	ACTION	Examine the ULOG for more information concerning the reason for the failure. Determine if the SNACRM process is still active.

5066:ERROR	**Encode SEND_DATA failed, RRCORR(%ld)!**	
	DESCRIPTION	An internal error has been detected by gw_nw_reply.
	ACTION	Contact BEA Customer Support.
5067:ERROR	**Decode inbound ACALL request failed!**	
	DESCRIPTION	An error has occurred while decoding an inbound ACALL request in gw_nw_decode.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
5068:ERROR	**Could not get FUNC**	
	DESCRIPTION	An internal error has been detected by gw_nw_acall.
	ACTION	Contact BEA Customer Support.
5069:ERROR	**Could not get FUNC**	
	DESCRIPTION	An internal error has been detected by gw_nw_connect.
	ACTION	Contact BEA Customer Support.
5070:ERROR	**SECURITY: Failed in security checking for RRCORR(%lx)!**	
	DESCRIPTION	Security checking failed for the inbound ACALL request. An error was returned by gw_nw_SetSecurity.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
5071:ERROR	**BAD Transaction ID shutdown**	
	DESCRIPTION	An internal error has been detected by gw_nw_SendData.
	ACTION	Contact BEA Customer Support.

5072:ERROR	**BAD Transaction ID shutdown**	
	DESCRIPTION	An internal error has been detected by gw_nw_OutBoundAcallRsp.
	ACTION	Contact BEA Customer Support.
5073:ERROR	**Could not get API**	
	DESCRIPTION	An internal error has been detected by gw_nw_QueryProcess.
	ACTION	Contact BEA Customer Support.
5074:ERROR	**Decode outbound ACALL response failed, RRCORR(%lx)!**	
	DESCRIPTION	An error has occurred while decoding the response from the host application to an outbound ACALL.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
5075:ERROR	**Invalid query_rsp index**	
	DESCRIPTION	An internal error has been detected by gw_nw_QueryProcessError.
	ACTION	Contact BEA Customer Support.
5076:ERROR	**Invalid connect_resp index**	
	DESCRIPTION	An internal error has been detected by gw_nw_ConnectError.
	ACTION	Contact BEA Customer Support.
5077:ERROR	**Enter shutdown**	
	DESCRIPTION	A failure has been detected while sending the SNACRM_QSVC_RSP verb to the SNACRM.
	ACTION	Examine the ULOG for more information concerning the reason for the failure. Determine if the SNACRM process is still active.

5078:ERROR	**Enter shutdown**	
	DESCRIPTION	A failure has been detected while sending the SNACRM_CONNECT_RSP verb to the SNACRM.
	ACTION	Examine the ULOG for more information concerning the reason for the failure. Determine if the SNACRM process is still active.
5079:ERROR	**BAD Transaction ID shutdown**	
	DESCRIPTION	An internal error has been detected by gw_nw_InBoundAcallRqst.
	ACTION	Contact BEA Customer Support.
5080:ERROR	**Decode inbound connect request failed, RRCORR(%lx)!**	
	DESCRIPTION	An error has occurred while decoding an inbound connect request in gw_nw_InBoundConnectRqst.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
5081:ERROR	**SECURITY: Inbound connect request security check failed, RRCORR(%lx)!**	
	DESCRIPTION	Security checking failed for the inbound CONNECT request. An error was returned by gw_nw_SetSecurity.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
5082:ERROR	**Decode SEND_DATA failed, RRCORR(%lx)!**	
	DESCRIPTION	An error has occurred while decoding an outbound send data request in gw_nw_SendData.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.

5083:ERROR	**Invalid buffer type, encode failed!**	
	DESCRIPTION	The buffer type specified for the outbound message was invalid. eAM gateway only supports standard Tuxedo buffer types over the domain gateway.
	ACTION	Correct the DMCONFIG definition.
5084:ERROR	**Outbound ACALL encode failed!**	
	DESCRIPTION	An error has occurred while encoding an outbound ACALL request in gw_nw_encode.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
5085:ERROR	**BAD SNACRM_IDX shutdown**	
	DESCRIPTION	An internal error has been detected by gw_nw_InBoundAcallRqst.
	ACTION	Contact BEA Customer Support.
5086:ERROR	**BAD Context shutdown**	
	DESCRIPTION	An internal error has been detected by gw_nw_InBoundAcallRqst.
	ACTION	Contact BEA Customer Support.
5087:ERROR	**Invalid buffer type, decode failed!**	
	DESCRIPTION	The buffer type specified for the inbound message was invalid. eAM gateway only supports standard Tuxedo buffer types over the domain gateway.
	ACTION	Correct the DMCONFIG definition.
5088:ERROR	**Unable to get RDOM**	
	DESCRIPTION	An internal error has been detected by gw_nw_init_env.
	ACTION	Contact BEA Customer Support.

5089:ERROR	**BAD SNACRM_IDX shutdown**	
	DESCRIPTION	An internal error has been detected by gw_nw_InBoundConnectRqst.
	ACTION	Contact BEA Customer Support.
5090:ERROR	**Outbound CONNECT encode failed!**	
	DESCRIPTION	An error has occurred while encoding an outbound CONNECT request in gw_nw_encode.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
5091:ERROR	**Outbound CONVSEND encode failed!**	
	DESCRIPTION	An error has occurred while encoding an outbound CONVSEND request in gw_nw_encode.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
5093:ERROR	**Bad Connect**	
	DESCRIPTION	An internal error has been detected by gw_nw_crmrcv.
	ACTION	Contact BEA Customer Support.
5094:ERROR	**Unable to send data on sna_conv_idx %d, gpnd = %d**	
	DESCRIPTION	An error has occurred while sending a connect response to the SNACRM.
	ACTION	Examine the ULOG for more information concerning the reason for the failure.
5095:ERROR	**BAD Transaction ID shutdown**	
	DESCRIPTION	An internal error has been detected by gw_nw_InBoundConnectRqst.
	ACTION	Contact BEA Customer Support.

5096:ERROR	**Unable to obtain socket id**	
	DESCRIPTION	An internal error has been detected by gw_nw_link_gp.
	ACTION	Contact BEA Customer Support.
5097:ERROR	**Unable to get RDOM**	
	DESCRIPTION	An internal error has been detected by gw_nw_link_gp.
	ACTION	Contact BEA Customer Support.
5098:ERROR	**Invalid action index**	
	DESCRIPTION	An internal error has been detected by gw_nw_link_gp.
	ACTION	Contact BEA Customer Support.
5099:ERROR	**Allocate context failed!**	
	DESCRIPTION	An internal error has been detected, an error was returned by gw_nw_AllocNwCtx.
	ACTION	Contact BEA Customer Support.
5100:ERROR	**Unable to obtain socket id**	
	DESCRIPTION	An internal error has been detected by gw_nw_init_env.
	ACTION	Contact BEA Customer Support.
5101:ERROR	**Unable to find view file for %s %s**	
	DESCRIPTION	Failed to find the view file for the FML buffer type specified in the INBUFTYPE for an inbound request.
	ACTION	Examine the environment variables VIEWDIR, VIEWFILES, VIEW32DIR, and VIEW32FILES to determine if the requested view is present.

5102:ERROR	**Invalid action index**	
	DESCRIPTION	An internal error has been detected by gw_nw_reply.
	ACTION	Contact BEA Customer Support.
5103:ERROR	**Invalid action index**	
	DESCRIPTION	An internal error has been detected by gw_nw_reply.
	ACTION	Contact BEA Customer Support.
5104:ERROR	**Invalid action index**	
	DESCRIPTION	An internal error has been detected by gw_nw_QueryProcess.
	ACTION	Contact BEA Customer Support.
5105:ERROR	**Invalid buffer pointer**	
	DESCRIPTION	An internal error has been detected by gw_nw_encode.
	ACTION	Contact BEA Customer Support.
5106:ERROR	**Allocate send buffer failed!**	
	DESCRIPTION	An internal error has been detected, an error was returned by gw_nw_empty_sndbuf.
	ACTION	Contact BEA Customer Support.
5107:ERROR	**Unable to get appkey for this outbound request**	
	DESCRIPTION	An internal error has been detected, gw_shm_getbylink returned an error while getting the remote domain entry.
	ACTION	Examine the DMCONFIG remote domain definitions and its security setting. Correct any errors found. If no error is found, contact BEA Customer Support.

5108:ERROR	**Enter shutdown**	
	DESCRIPTION	An internal error has been detected by gw_nw_SendEvent.
	ACTION	Contact BEA Customer Support.
5109:ERROR	**Unable to find view file for %s %s**	
	DESCRIPTION	Failed to find the view file for encoding outbound FML data as specified in the OUTBUFTYPE.
	ACTION	Examine the environment variables VIEWDIR, VIEWFILES, VIEW32DIR, and VIEW32FILES to determine if the requested view is present.
5110:ERROR	**Unable to find view file for %s %s**	
	DESCRIPTION	Failed to find the view file for encoding outbound FML32 data as specified in the OUTBUFTYPE.
	ACTION	Examine the environment variables VIEWDIR, VIEWFILES, VIEW32DIR, and VIEW32FILES to determine if the requested view is present.
5111:ERROR	**BAD Context shutdown caused at transaction associate**	
	DESCRIPTION	An internal error has been detected, an error was returned by gw_nw_AssociateTranByTCTXT.
	ACTION	Contact BEA Customer Support.
5112:ERROR	**System full, can't create action, Enter shutdown**	
	DESCRIPTION	Failed to create GWSNAX gateway internal scheduled event to handle failure ore disconnect.
	ACTION	Either memory allocation failed or a system limit was reached. Contact BEA Customer Support.
5113:ERROR	**Realloc buffer of size(%d) failed!**	
	DESCRIPTION	An internal error has been detected, gw_nw_Realloc was unable to allocate memory needed.
	ACTION	Contact BEA Customer Support.

5114:ERROR	**Invalid network descriptor(%d)!**	
	DESCRIPTION	An internal error has been detected by gw_nw_acall.
	ACTION	Contact BEA Customer Support.
5115:ERROR	**Encryption setup failed, code %d**	
	DESCRIPTION	An error has occurred while establishing link-level encryption with the SNACRM. -1:The processes are using different encryption types (ex. GPE) 6: The processes are using different versions of the encryption protocol. 7: The negotiation of an encryption level has failed. 8: Asystem error has occurred in the encryption setup
	ACTION	Verify that the setup of encryption on both sides of the link is correct. Verify that there are common encryption levels in the ranges specified on the process command lines. Verify that the correct encryption libraries are installed.
5116:ERROR	**SNACRM authentication setup failed**	
	DESCRIPTION	An error has occurred while authenticating a SNACRM client.
	ACTION	Verify that the authentication file(s) for both the SNACRM and the client are set up correctly, and that both have correct access privileges. Verify that the authentication file is specified correctly on both command lines.

5117:INFO	**Link encryption established, %s bits**	
	DESCRIPTION	The SNAX has established link-level encryption with the SNACRM..
	ACTION	No action is required. If the encryption level specidied does not match the desired value, verify the encryption level range specified on the SNAX and SNACRM command lines. A value of ‘40/56’ indicates that encryption is set at 56, even though the specified range may be as low as 40.
6000:ERROR	**YOUR eLink SNA LICENSE IS EITHER INVALID OR EXPIRED**	
	DESCRIPTION	The eLink for Mainframe SNA v3 license is missing from the lic.txt file in $TUXDIR/udataobj or it has expired.
	ACTION	Look in the $TUXDIR/udataobj/lic.txt file for a section labeled eLink_SNA v3 (or CONNECT_SNA 3.x) and if present, check the expiration date.
9001:ERROR	**<taskname> timed out with failCode <failcode>**	
	DESCRIPTION	A conversation has timed out in the SNACRM with the stack return code of <failcode>. A timer event set to watch a conversation has expired. <taskname> may appear as: OB-Conversation #*nn* (<linkref>) tx #*m* <tranname>, or IB-Conversation #nn (<linkref>) tx #*m* <tranname> where: *nn* is an internal APPC conversation number. *m* is the transaction context where -1 signifies non-transactional.
	ACTION	Examine stderr and the ULOG for additional information concerning the failure.

9002:ERROR	**Server (<stackref>) Creation Failed**	
	DESCRIPTION	SNACRM was unable to instantiate the stack object due to an error.
	ACTION	Check for additional messages in stderr. Could be that the shared library for the stack or the stack interface could not be loaded due to an incorrect library path.
9003:ERROR	**Server Failed (<stackref>), Code = <returncode>**	
	DESCRIPTION	SNACRM received a bad return code from the stack start-up.
	ACTION	The <returncode> is the value returned by the SNA Stack software. Check the status of the stack and the configuration of the stack and the DMCONFIG. **Note:** Unless you started the SNACRM as a Tuxedo server, you must manually kill the SNACRM process.
9004:ERROR	**Configuration change on link <linkref> requires cold start**	
	DESCRIPTION	Attempting to do a warm start after changing the domain configuration.
	ACTION	Change start type to "COLD" and restart.
9005:WARNING	**Unrecovered transaction, ID=<tctxt>, blob dropped. Transaction presumed forgotten.**	
	DESCRIPTION	An attempt was make by Tuxedo to recover the specified transaction which was unknown by the SNACRM. It is presumed that it was already committed or aborted prior to the recovery attempt.
	ACTION	None. This message for information only.

9006:ERROR	**Unable to start the recovery task for link <linkref>**	
	DESCRIPTION	An error occurred during the warm start of Tuxedo.
	ACTION	Cold start the Tuxedo application.
9008:WARNING	**Unknown tranid dropped, id=<tctxt>**	
	DESCRIPTION	Recovery was requested by Tuxedo on a transaction that was already forgotten by the SNACRM.
	ACTION	None. This message for information only.
9009:ERROR	**No blob with recovery request. Transaction dropped, id=<tctxt>**	
	DESCRIPTION	Attempting to warm start after the SNACRM's BLOBLOG has been modified.
	ACTION	Change start type to "COLD" and restart.

9010:ERROR	**<taskname> failed with failCode <failcode>**	
	DESCRIPTION	A conversation has failed with the stack return code of <failcode>. <taskname> may appear as: OB-Conversation #*nn* (<linkref>) tx #*m* <tranname>, or IB-Conversation #nn (<linkref>) tx #*m* <tranname> where: *nn* is an internal APPC conversation number. *m* is the transaction context where -1 signifies non-transactional. Possible values for the <failcode> are: 1. Communications - unable to create the APPCserver object. 2. MemoryAllocation - internal error allocating memory. 3. InvalidObject - a SNACRM object could not be created or has been made invalid by some previous error. 4. InputOutput - error occurred during file I/O or an unexpected APPC return code was received. 5. Registration - internal task cannot be registered.
	ACTION	Examine stderr and the ULOG for additional information concerning the failure. For failcode InputOutput, verify that the user starting the SNACRM process has the proper file permissions for the BLOBLOG and RSTRTLOG. If no apparent error is found, contact BEA Customer Support.

9011:ERROR	**Attempt to connect as second master refused!**	
	DESCRIPTION	A second eAM gateway is attempting to connect to the CRM as a master gateway. Only one master gateway is allowed.
	ACTION	Ensure that multiple Tuxedo configurations do not use the same SNACRM address.
9012:ERROR	**Attempt to connect as master in autonomous mode refused!**	
	DESCRIPTION	An attempt to connect to the SNACRM as a master gateway was made when the SNACRM was running in autonomous mode.
	ACTION	Ensure that multiple Tuxedo configurations do not use the same SNACRM address.
9013:ERROR	**Attempt to connect with incorrect group name (<groupname>) refused!**	
	DESCRIPTION	The group name in the DMCONFIG file does not match the group name specified in the SNACRM command line.
	ACTION	Correct the group name that is in error and restart.
9014:ERROR	**INTERNAL ERROR: memory allocation failed [for new context|data buffer]**	
	DESCRIPTION	Internal error allocating memory.
	ACTION	Contact BEA Customer Support.
9015:ERROR	**INTERNAL ERROR: server registration failed**	
	DESCRIPTION	Internal error registering the APPC server.
	ACTION	Contact BEA Customer Support.
9016:ERROR	**Link refers to undefined APPC stack (<stackref>)!**	
	DESCRIPTION	The stackref in the link configuration is incorrect.
	ACTION	Correct the stackref that is in error, run dmloadcf, and restart.

9017:ERROR	**INTERNAL ERROR: link registration failed**	
	DESCRIPTION	Internal error registering the link.
	ACTION	Contact BEA Customer Support.
9018:ERROR	**Invalid Transition <additional information>**	
	DESCRIPTION	An internal state table failure has occurred. The <additional information> will be one of the following: 1. From <oldstate> to <newstate> for <dir>-bound transaction #<tid> 2. To <newstate> for in-bound transaction #<tid> 3. To <newstate> for out-bound transaction #<tid>
	ACTION	Contact BEA Customer Support.
9019:ERROR	**Unknown Service Correlator = <correlator>, message dropped**	
	DESCRIPTION	Internal error assigning service correlator values.
	ACTION	Contact BEA Customer Support.
9020:ERROR	**Duplicate Service Correlator = <correlator>**	
	DESCRIPTION	Internal error assigning service correlator values.
	ACTION	Contact BEA Customer Support.
9021:ERROR	**Invalid Remote Link Name <linkref>**	
	DESCRIPTION	The remote link name in a request does not match any defined link name.
	ACTION	Correct the DMCONFIG and restart.
9022:ERROR	**Invalid transaction context = <tctxt>**	
	DESCRIPTION	Internal error assigning transaction context values.
	ACTION	Contact BEA Customer Support.

9023:ERROR	**Unknown Service Correlator = <correlator>, message dropped**	
	DESCRIPTION	Internal error assigning service correlator values.
	ACTION	Contact BEA Customer Support.
9024:ERROR	**Invalid initial syncpoint received from subordinate, tx#<tid>**	
	DESCRIPTION	Syncpoint processing protocol violation. Subordinate member of conversation attempted to initiate a syncpoint.
	ACTION	Contact BEA Customer Support.
9025:ERROR	**Invalid Input Message Discarded**	
	DESCRIPTION	Internal error, bad message sent between eAM gateway and SNACRM.
	ACTION	Contact BEA Customer Support.
9026:ERROR	**CNOS Notification Received for unknown partner <partnerLU>**	
	DESCRIPTION	Multiple instances of the SNACRM may be using the same local LU.
	ACTION	Ensure that multiple Tuxedo configurations do not use the same local LU.
9027:WARNING	**Remote Stop Received for <linkref>**	
	DESCRIPTION	The remote host has issued a stop for the specified link.
	ACTION	None. This message for information only.
9028:WARNING	**Remote Start Received for <linkref>**	
	DESCRIPTION	The remote host has issued a start for the specified link.
	ACTION	None. This message for information only.

9029:ERROR	**Undefined Remote LU on link <linkref>**	
	DESCRIPTION	The remote LU does not exist as defined.
	ACTION	Check the DMCONFIG file and the stack configuration and correct the mis-match.
9030:ERROR	**Unable to start session on link <linkref>. Reason=<reason>**	
	DESCRIPTION	Link activation failure due to SNA error.
	ACTION	<reason> is the description of the stack return code. Determine the cause and correct.
9031:ERROR	**Unable to initialize link <linkref>. Reason=<reason>**	
	DESCRIPTION	Link initialization failure due to SNA error.
	ACTION	<reason> is the description of the stack return code. Determine the cause and correct.
9032:ERROR	**No Available Session on link <linkref> for context <correlator>**	
	DESCRIPTION	Max sessions has been exceeded.
	ACTION	Check session limits in DMCONFIG, stack configuration, CICS or VTAM. Increase if necessary.
9033:ERROR	**Requested Synclevel not supported by link <linkref> for context <correlator> (synclevel <level>)**	
	DESCRIPTION	Attempted to issues a request at sync level <level> on a link that does not support that level.
	ACTION	Correct application or DMCONFIG.
9034:ERROR	**Service Request at SyncLevel=2 Rejected on PENDING link <linkref> for context <correlator>**	
	DESCRIPTION	An attempt to start a new sync level 2 request has been received and the Link is currently processing recovery information.
	ACTION	Wait until recovery is complete to request sync level 2 services.

9035:ERROR	**Inbound Request Transform Failed (<status>) for context <correlator>**	
	DESCRIPTION	An error has occurred while processing the CICS transform for an inbound DPL request. This normally occurs when the API entry in the DMCONFIG for the local service specifies CICS instead of ATMI.
	ACTION	Check DMCONFIG for incorrect specification of local service API entry.
9036:ERROR	**Inbound Response Transform Failed (<status>) for context <correlator>**	
	DESCRIPTION	An error has occurred while processing the CICS transform for an inbound DPL response. This normally occurs when the API entry in the DMCONFIG for the local service specifies CICS instead of ATMI.
	ACTION	Check DMCONFIG for incorrect specification of local service API entry.
9037:ERROR	**Outbound Request Transform Failed (<status>) for context <correlator>**	
	DESCRIPTION	An error has occurred while processing the CICS transform for an outbound DPL request. This normally occurs when the API entry in the DMCONFIG for the remote service specifies CICS instead of ATMI.
	ACTION	Check DMCONFIG for incorrect specification of local service API entry.

9038:ERROR	**Outbound Response Transform Failed (<status>) for context <correlator>**	
	DESCRIPTION	An error has occurred while processing the CICS transform for an outbound DPL response. This normally occurs when the API entry in the DMCONFIG for the remote service specifies CICS instead of ATMI.
	ACTION	Check DMCONFIG for incorrect specification of local service API entry.
9039:ERROR	**Conversation terminated without confirm for context <correlator>**	
	DESCRIPTION	Sync level 2 conversation was terminated with out confirm.
	ACTION	Check application program and correct.
9040:ERROR	**Inbound Confirm not supported**	
	DESCRIPTION	Host application is requesting an inbound confirm. This is not supported.
	ACTION	Check host application program and correct.
9041:ERROR	**Inbound Confirm for multi-ISRT not supported**	
	DESCRIPTION	Host IMS application is requesting an inbound confirm and using multiple ISRT commands. This is not supported.
	ACTION	Check host application program and correct.
9043:ERROR	**Missing send last from host (ATMI request/response) for context <correlator>**	
	DESCRIPTION	Host application did not issue send last during an out-bound request/response service. The host application may have abended.
	ACTION	Check application program and correct.

9044:INFO	**DPL program abended with CICS code <abendcode>, program=<progname>**	
	DESCRIPTION	The specified host DPL program has abended with the code specified.
	ACTION	None. This message is for information only.
9045:INFO	**DPL program failed with CICS rcode <eibrcode>, program=<progname>**	
	DESCRIPTION	The specified host DPL program has failed with the eibrcode specified.
	ACTION	None. This message is for information only.
9046:ERROR	**Invalid combination for Service Context <correlator>, <combination>**	
	DESCRIPTION	The specified <combination> is invalid. It will be one of the following: 1. Sync-Level, function, and API 2. Function and API
	ACTION	Examine the DMCONFIG and make corrections.
9047:ERROR	**Sequence number error for Service Context <correlator>, seqno <seqno>**	
	DESCRIPTION	There has been a sequence number failure for the specified context.
	ACTION	Contact BEA Customer Support.
9048:ERROR	**Invalid conversation task for Service Context <correlator>, task=<task>**	
	DESCRIPTION	The conversation has already been terminated.
	ACTION	Contact BEA Customer Support.

9049:ERROR	**Invalid task switch for Service Context <correlator>, from <task1> to <task2>**	
	DESCRIPTION	An internal protocol violation has occurred.
	ACTION	Contact BEA Customer Support.
9050:ERROR	**Transformer creation failed for in-bound transaction <trancode>**	
	DESCRIPTION	An internal error has occurred.
	ACTION	Contact BEA Customer Support.
9051:ERROR	**Transformer failed for in-bound transaction <trancode>**	
	DESCRIPTION	An internal error has occurred. Resource name is not present.
	ACTION	Contact BEA Customer Support.
9052:WARNING	**Inter-task Message dropped (<verbname>), parm=<parm> From: <task1> to <task2>**	
	DESCRIPTION	An internal message between two tasks has been dropped.
	ACTION	None. This message is for information only.
9053:ERROR	**Attempt to send <nnnnn> bytes (> 32767)**	
	DESCRIPTION	The length of a send request exceeded 32767 (including overhead).
	ACTION	Check application program and correct.
9054:ERROR	**Allocation Failure for <trancode> on <remotesysid>: <error>**	
	DESCRIPTION	An Allocation error occurred.
	ACTION	The reason for the failure is described by <error>. Correct problem with configuration or application.

9055:ERROR	**Invalid Exchange Logs GDS variable received from <remotesysid>**	
	DESCRIPTION	The log files for the SNACRM have been incorrectly modified.
	ACTION	Run CRMLOGS to examine the SNACRM log file. Cold start the Tuxedo application.
9056:ERROR	**Invalid cold start received from <remotesysid>. Unrecovered local transactions are pending.**	
	DESCRIPTION	Attempting to cold start host while warm starting Tuxedo.
	ACTION	Run CRMLOGS to examine the SNACRM log file. Cold start the Tuxedo application.
9057:ERROR	**Invalid warm start received from <remotesysid>. Unknown log name.**	
	DESCRIPTION	The log files for the SNACRM have been incorrectly modified.
	ACTION	Run CRMLOGS to examine the SNACRM log file. Cold start the Tuxedo application.
9058:ERROR	**Invalid Compare States GDS variable received from <remotesysid>**	
	DESCRIPTION	The log files for the SNACRM have been incorrectly modified.
	ACTION	Run CRMLOGS to examine the SNACRM log file. Cold start the Tuxedo application.
9059:ERROR	**Mixed Heuristic on link <linkref> for <unitofwork> Correlator [<correlator>]**	
	DESCRIPTION	One side has reported committed while the other side has reported aborted.
	ACTION	Check the ULOG for any additional messages.

9060:WARNING	**Inbound Exchange Logs Rejected for <remotesysid>**	
	DESCRIPTION	Link not configured for sync level 2.
	ACTION	None. This message is for information only.
9061:WARNING	**Link <linkref> not configured for synclevel 2**	
	DESCRIPTION	Link specified by <linkref> is not configured for sync level 2.
	ACTION	None. This message is for information only.
9062:ERROR	**Exchange Logs Rejected for <remotesysid>, Restart Type or Log Name Mismatch**	
	DESCRIPTION	The log files for the SNACRM have been incorrectly modified.
	ACTION	Run CRMLOGS to examine the SNACRM log file. Cold start the Tuxedo application.
9063:ERROR	**Exchange Logs failed with <linkref>**	
	DESCRIPTION	An error occurred during the exchange logs process.
	ACTION	Run CRMLOGS to examine the SNACRM log file. Cold start the Tuxedo application.
9064:ERROR	**Invalid initial syncpoint received from subordinate, %s**	
	DESCRIPTION	An internal error has occurred during the commit process.
	ACTION	Contact BEA Customer Support
9069: ERROR	**SNACRM encryption setup failed**	
	DESCRIPTION	An error has occurred while establishing link-level encryption with the SNACRM.
	ACTION	Verify that the setup of encryption on both sides of the link is correct. Verify that there are common encryption levels in the ranges specified on the process command lines. Verify that the correct encryption libraries are installed

9072: ERROR	**Attempted access by unauthorized SNACRM client**	
	DESCRIPTION	A client has attempted to access the SNACRM without the proper authentication or encryption setup.
	ACTION	Verify that the client should have access to the SNACRM. Verify that encryption is set up correctly in both the SNACRM and the client, and that the correct security add-on packages are installed. Verify that the authentication file is set up correctly, and that both the CRM and the client have correct access privileges.
9073: ERROR	**SNACRM authentication setup failed**	
	DESCRIPTION	An error has occurred while authenticating a SNACRM client.
	ACTION	Verify that the authentication file(s) for both the SNACRM and the client are set up correctly, and that both have correct access privileges. Verify that the authentication file is specified correctly on both command lines.

C Code Page Translation Tables

This section explains how to modify Code Page Translation Tables and shows examples of the Code Page Translation Tables that are provided with BEA eLink Adapter for Mainframe software. The files actually containing these tables are located in the `$TUXDIR/udataobj/codepage` sub-directory on your product CDROM.

This section discusses the following topics:

Modifying a Code Page Translation Table

The tables provide conversions between the ASCII Latin-1 character set and representative national language EBCDIC character sets. In most cases, you do not have to modify them. Simply choose the appropriate translation table for a selected language and enter its file name in the CODEPAGE specification, as explained in "Data Translations".

However, if you must modify a translation table to suit your purpose, be aware of the following:

- Make sure you have character mapping information and that you know which code represents a given character. This information is available from a number of sources and is not provided in this documentation. A good source is the *IBM National Language Support Reference Manual*.
- If you modify a character code in an outbound table, you must also modify its inbound counterpart.
- Build tables from scratch is not recommended.
- The tables have a common format that contains comment lines and required lines. The format must be maintained to ensure proper table operation. Comment lines begin with the # character. ***Do not alter*** the following required lines:
  - `version (100)` specifies the format of the rest of the file.
  - `table (256)` specifies the size of the table and the min/max number of bytes composing each character code.

To modify a table, perform the following steps:

1. Open the file you want to modify with the text editor of your choice. For example:

   ```
   edit $TUXDIR/udataobj/codepage/00819x00273
   ```

   The text editor opens the file, in this example the translation tables for Germany (00819x00273).
2. Modify the character code in the outbound table, using the editor functions.
3. Modify the counterpart character code in the inbound table, using the editor functions.

4. Repeat Steps 2 and 3 until you have completed the modifications.
5. Save the file, using the editor functions. Be sure to give it a name other than the original. ***Do not save modifications to any of the original files*** provided with your product CD ROM.

Note: To use the file you modified for code page translations, make sure you specify its name using the `CODEPAGE` option in the `DM_REMOTE_DOMAINS` section of the gateway `DMCONFIG` file.

6. Exit the editor.

Default Tuxedo Code Page Translation Table

```
#================================================================
# tuxedo
#Default Tuxedo ASCII/EBCDIC character translation tables.
#
# Local:    "TUXEDO-ASCII"
# Remote:   "TUXEDO-EBCDIC"
# Built:    1999-04-13 22:12:00 UT
#
# @(#)$Id: tuxedo,v 1.1 1999/04/16 20:08:09 david Exp $
#----------------------------------------------------------------

# Header
version 100;

# Outbound (local -> remote) table
table 256 1 1;
 00 01 02 03  37 2D 2E 2F  16 05 25 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  3C 3D 32 26  18 19 3F 27  1C 1D 1E 1F  # 10-1F
 40 5A 7F 7B  5B 6C 50 7D  4D 5D 5C 4E  6B 60 4B 61  # 20-2F
 F0 F1 F2 F3  F4 F5 F6 F7  F8 F9 7A 5E  4C 7E 6E 6F  # 30-3F
 7C C1 C2 C3  C4 C5 C6 C7  C8 C9 D1 D2  D3 D4 D5 D6  # 40-4F
 D7 D8 D9 E2  E3 E4 E5 E6  E7 E8 E9 AD  E0 BD 5F 6D  # 50-5F
 79 81 82 83  84 85 86 87  88 89 91 92  93 94 95 96  # 60-6F
 97 98 99 A2  A3 A4 A5 A6  A7 A8 A9 C0  6A D0 A1 07  # 70-7F
 20 21 22 23  24 15 06 17  28 29 2A 2B  2C 09 0A 1B  # 80-8F
 30 31 1A 33  34 35 36 08  38 39 3A 3B  04 14 3E E1  # 90-9F
 41 42 43 44  45 46 47 48  49 51 52 53  54 55 56 57  # A0-AF
 58 59 62 63  64 65 66 67  68 69 70 71  72 73 74 75  # B0-BF
 76 77 78 80  8A 8B 8C 8D  8E 8F 90 9A  9B 9C 9D 9E  # C0-CF
 9F A0 AA AB  AC 4A AE AF  B0 B1 B2 B3  B4 B5 B6 B7  # D0-DF
 B8 B9 BA BB  BC 4F BE BF  CA CB CC CD  CE CF DA DB  # E0-EF
 DC DD DE DF  EA EB EC ED  EE EF FA FB  FC FD FE FF  # F0-FF

# Inbound (remote -> local) table
table 256 1 1;
 00 01 02 03  9C 09 86 7F  97 8D 8E 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  9D 85 08 87  18 19 92 8F  1C 1D 1E 1F  # 10-1F
 80 81 82 83  84 0A 17 1B  88 89 8A 8B  8C 05 06 07  # 20-2F
 90 91 16 93  94 95 96 04  98 99 9A 9B  14 15 9E 1A  # 30-3F
 20 A0 A1 A2  A3 A4 A5 A6  A7 A8 D5 2E  3C 28 2B E5  # 40-4F
 26 A9 AA AB  AC AD AE AF  B0 B1 21 24  2A 29 3B 5E  # 50-5F
 2D 2F B2 B3  B4 B5 B6 B7  B8 B9 7C 2C  25 5F 3E 3F  # 60-6F
 BA BB BC BD  BE BF C0 C1  C2 60 3A 23  40 27 3D 22  # 70-7F
 C3 61 62 63  64 65 66 67  68 69 C4 C5  C6 C7 C8 C9  # 80-8F
 CA 6A 6B 6C  6D 6E 6F 70  71 72 CB CC  CD CE CF D0  # 90-9F
 D1 7E 73 74  75 76 77 78  79 7A D2 D3  D4 5B D6 D7  # A0-AF
 D8 D9 DA DB  DC DD DE DF  E0 E1 E2 E3  E4 5D E6 E7  # B0-BF
 7B 41 42 43  44 45 46 47  48 49 E8 E9  EA EB EC ED  # C0-CF
 7D 4A 4B 4C  4D 4E 4F 50  51 52 EE EF  F0 F1 F2 F3  # D0-DF
 5C 9F 53 54  55 56 57 58  59 5A F4 F5  F6 F7 F8 F9  # E0-EF
 30 31 32 33  34 35 36 37  38 39 FA FB  FC FD FE FF  # F0-FF

# End
```

United States (00819x00037) Code Page Translation Table

```
#================================================================
# 00819x00037
#Character code page mapping tables for US (USA).
#
# Local:    "IBM-CP00819", ISO-8859-1 Latin-1
# Remote:   "IBM-CP00037", EBCDIC Latin-1, US
# Built:    1999-04-16 19:50:00 UT
#
# @(#)$Id: 00819x00037,v 1.3.2.1 1999/04/29 13:03:56 cmadm Exp $
#----------------------------------------------------------------

# Header
version 100;
```

```
# Outbound (local -> remote) table
table 256 1 1;
 00 01 02 03  37 2D 2E 2F  16 05 15 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  3C 3D 32 26  18 19 3F 27  1C 1D 1E 1F  # 10-1F
 40 5A 7F 7B  5B 6C 50 7D  4D 5D 5C 4E  6B 60 4B 61  # 20-2F
 F0 F1 F2 F3  F4 F5 F6 F7  F8 F9 7A 5E  4C 7E 6E 6F  # 30-3F
 7C C1 C2 C3  C4 C5 C6 C7  C8 C9 D1 D2  D3 D4 D5 D6  # 40-4F
 D7 D8 D9 E2  E3 E4 E5 E6  E7 E8 E9 BA  E0 BB B0 6D  # 50-5F
 79 81 82 83  84 85 86 87  88 89 91 92  93 94 95 96  # 60-6F
 97 98 99 A2  A3 A4 A5 A6  A7 A8 A9 C0  4F D0 A1 07  # 70-7F
 04 06 08 09  0A 14 17 1A  1B 20 21 22  23 24 25 28  # 80-8F
 29 2A 2B 2C  30 31 33 34  35 36 38 39  3A 3B 3E FF  # 90-9F
 41 AA 4A B1  9F B2 6A B5  BD B4 9A 8A  5F CA AF BC  # A0-AF
 90 8F EA FA  BE A0 B6 B3  9D DA 9B 8B  B7 B8 B9 AB  # B0-BF
 64 65 62 66  63 67 9E 68  74 71 72 73  78 75 76 77  # C0-CF
 AC 69 ED EE  EB EF EC BF  80 FD FE FB  FC AD AE 59  # D0-DF
 44 45 42 46  43 47 9C 48  54 51 52 53  58 55 56 57  # E0-EF
 8C 49 CD CE  CB CF CC E1  70 DD DE DB  DC 8D 8E DF  # F0-FF

# Inbound (remote -> local) table
table 256 1 1;
 00 01 02 03  80 09 81 7F  82 83 84 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  85 0A 08 86  18 19 87 88  1C 1D 1E 1F  # 10-1F
 89 8A 8B 8C  8D 8E 17 1B  8F 90 91 92  93 05 06 07  # 20-2F
 94 95 16 96  97 98 99 04  9A 9B 9C 9D  14 15 9E 1A  # 30-3F
 20 A0 E2 E4  E0 E1 E3 E5  E7 F1 A2 2E  3C 28 2B 7C  # 40-4F
 26 E9 EA EB  E8 ED EE EF  EC DF 21 24  2A 29 3B AC  # 50-5F
 2D 2F C2 C4  C0 C1 C3 C5  C7 D1 A6 2C  25 5F 3E 3F  # 60-6F
 F8 C9 CA CB  C8 CD CE CF  CC 60 3A 23  40 27 3D 22  # 70-7F
 D8 61 62 63  64 65 66 67  68 69 AB BB  F0 FD FE B1  # 80-8F
 B0 6A 6B 6C  6D 6E 6F 70  71 72 AA BA  E6 B8 C6 A4  # 90-9F
 B5 7E 73 74  75 76 77 78  79 7A A1 BF  D0 DD DE AE  # A0-AF
 5E A3 A5 B7  A9 A7 B6 BC  BD BE 5B 5D  AF A8 B4 D7  # B0-BF
 7B 41 42 43  44 45 46 47  48 49 AD F4  F6 F2 F3 F5  # C0-CF
 7D 4A 4B 4C  4D 4E 4F 50  51 52 B9 FB  FC F9 FA FF  # D0-DF
 5C F7 53 54  55 56 57 58  59 5A B2 D4  D6 D2 D3 D5  # E0-EF
 30 31 32 33  34 35 36 37  38 39 B3 DB  DC D9 DA 9F  # F0-FF

# End
```

Germany (00819x00273) Code Page Translation Table

```
#================================================================
# 00819x00273
#Character code page mapping tables for Germany (Deutschland).
#
# Local:    "IBM-CP00819", ISO-8859-1 Latin-1
# Remote:   "IBM-CP00273", EBCDIC Latin-1, Germany
# Built:    1999-04-16 21:00:00 UT
#
# @(#)$Id: 00819x00273,v 1.4.2.1 1999/04/29 13:04:18 cmadm Exp $
#----------------------------------------------------------------

# Header
version 100;

# Outbound (local -> remote) table
table 256 1 1;
 00 01 02 03  37 2D 2E 2F  16 05 15 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  3C 3D 32 26  18 19 3F 27  1C 1D 1E 1F  # 10-1F
 40 4F 7F 7B  5B 6C 50 7D  4D 5D 5C 4E  6B 60 4B 61  # 20-2F
 F0 F1 F2 F3  F4 F5 F6 F7  F8 F9 7A 5E  4C 7E 6E 6F  # 30-3F
 B5 C1 C2 C3  C4 C5 C6 C7  C8 C9 D1 D2  D3 D4 D5 D6  # 40-4F
 D7 D8 D9 E2  E3 E4 E5 E6  E7 E8 E9 63  EC FC 5F 6D  # 50-5F
 79 81 82 83  84 85 86 87  88 89 91 92  93 94 95 96  # 60-6F
 97 98 99 A2  A3 A4 A5 A6  A7 A8 A9 43  BB DC 59 07  # 70-7F
 04 06 08 09  0A 14 17 1A  1B 20 21 22  23 24 25 28  # 80-8F
 29 2A 2B 2C  30 31 33 34  35 36 38 39  3A 3B 3E FF  # 90-9F
 41 AA B0 B1  9F B2 CC 7C  BD B4 9A 8A  BA CA AF BC  # A0-AF
 90 8F EA FA  BE A0 B6 B3  9D DA 9B 8B  B7 B8 B9 AB  # B0-BF
 64 65 62 66  4A 67 9E 68  74 71 72 73  78 75 76 77  # C0-CF
 AC 69 ED EE  EB EF E0 BF  80 FD FE FB  5A AD AE A1  # D0-DF
 44 45 42 46  C0 47 9C 48  54 51 52 53  58 55 56 57  # E0-EF
 8C 49 CD CE  CB CF 6A E1  70 DD DE DB  D0 8D 8E DF  # F0-FF

# Inbound (remote -> local) table
table 256 1 1;
 00 01 02 03  80 09 81 7F  82 83 84 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  85 0A 08 86  18 19 87 88  1C 1D 1E 1F  # 10-1F
 89 8A 8B 8C  8D 8E 17 1B  8F 90 91 92  93 05 06 07  # 20-2F
 94 95 16 96  97 98 99 04  9A 9B 9C 9D  14 15 9E 1A  # 30-3F
 20 A0 E2 7B  E0 E1 E3 E5  E7 F1 C4 2E  3C 28 2B 21  # 40-4F
 26 E9 EA EB  E8 ED EE EF  EC 7E DC 24  2A 29 3B 5E  # 50-5F
 2D 2F C2 5B  C0 C1 C3 C5  C7 D1 F6 2C  25 5F 3E 3F  # 60-6F
 F8 C9 CA CB  C8 CD CE CF  CC 60 3A 23  A7 27 3D 22  # 70-7F
 D8 61 62 63  64 65 66 67  68 69 AB BB  F0 FD FE B1  # 80-8F
 B0 6A 6B 6C  6D 6E 6F 70  71 72 AA BA  E6 B8 C6 A4  # 90-9F
 B5 DF 73 74  75 76 77 78  79 7A A1 BF  D0 DD DE AE  # A0-AF
 A2 A3 A5 B7  A9 40 B6 BC  BD BE AC 7C  AF A8 B4 D7  # B0-BF
 E4 41 42 43  44 45 46 47  48 49 AD F4  A6 F2 F3 F5  # C0-CF
 FC 4A 4B 4C  4D 4E 4F 50  51 52 B9 FB  7D F9 FA FF  # D0-DF
 D6 F7 53 54  55 56 57 58  59 5A B2 D4  5C D2 D3 D5  # E0-EF
 30 31 32 33  34 35 36 37  38 39 B3 DB  5D D9 DA 9F  # F0-FF

# End
```

Finland/Sweden (00819x00278) Code Page Translation Table

```
#================================================================
# 00819x00278
#      Character code page mapping tables for Finland/Sweden.
#
# Local:    "IBM-CP00819", ISO-8859-1 Latin-1
# Remote:   "IBM-CP00278", EBCDIC Latin-1, Finland/Sweden
# Built:    1999-04-16 19:50:00 UT
#
# @(#)$Id: 00819x00278,v 1.4.2.1 1999/04/29 13:04:01 cmadm Exp $
#----------------------------------------------------------------

# Header
version 100;

# Outbound (local -> remote) table
table 256 1 1;
 00 01 02 03  37 2D 2E 2F  16 05 15 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  3C 3D 32 26  18 19 3F 27  1C 1D 1E 1F  # 10-1F
 40 4F 7F 63  67 6C 50 7D  4D 5D 5C 4E  6B 60 4B 61  # 20-2F
 F0 F1 F2 F3  F4 F5 F6 F7  F8 F9 7A 5E  4C 7E 6E 6F  # 30-3F
 EC C1 C2 C3  C4 C5 C6 C7  C8 C9 D1 D2  D3 D4 D5 D6  # 40-4F
 D7 D8 D9 E2  E3 E4 E5 E6  E7 E8 E9 B5  71 9F 5F 6D  # 50-5F
 51 81 82 83  84 85 86 87  88 89 91 92  93 94 95 96  # 60-6F
 97 98 99 A2  A3 A4 A5 A6  A7 A8 A9 43  BB 47 DC 07  # 70-7F
 04 06 08 09  0A 14 17 1A  1B 20 21 22  23 24 25 28  # 80-8F
 29 2A 2B 2C  30 31 33 34  35 36 38 39  3A 3B 3E FF  # 90-9F
 41 AA B0 B1  5A B2 CC 4A  BD B4 9A 8A  BA CA AF BC  # A0-AF
 90 8F EA FA  BE A0 B6 B3  9D DA 9B 8B  B7 B8 B9 AB  # B0-BF
 64 65 62 66  7B 5B 9E 68  74 E0 72 73  78 75 76 77  # C0-CF
 AC 69 ED EE  EB EF 7C BF  80 FD FE FB  FC AD AE 59  # D0-DF
 44 45 42 46  C0 D0 9C 48  54 79 52 53  58 55 56 57  # E0-EF
 8C 49 CD CE  CB CF 6A E1  70 DD DE DB  A1 8D 8E DF  # F0-FF

# Inbound (remote -> local) table
table 256 1 1;
 00 01 02 03  80 09 81 7F  82 83 84 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  85 0A 08 86  18 19 87 88  1C 1D 1E 1F  # 10-1F
 89 8A 8B 8C  8D 8E 17 1B  8F 90 91 92  93 05 06 07  # 20-2F
 94 95 16 96  97 98 99 04  9A 9B 9C 9D  14 15 9E 1A  # 30-3F
 20 A0 E2 7B  E0 E1 E3 7D  E7 F1 A7 2E  3C 28 2B 21  # 40-4F
 26 60 EA EB  E8 ED EE EF  EC DF A4 C5  2A 29 3B 5E  # 50-5F
 2D 2F C2 23  C0 C1 C3 24  C7 D1 F6 2C  25 5F 3E 3F  # 60-6F
 F8 5C CA CB  C8 CD CE CF  CC E9 3A C4  D6 27 3D 22  # 70-7F
 D8 61 62 63  64 65 66 67  68 69 AB BB  F0 FD FE B1  # 80-8F
 B0 6A 6B 6C  6D 6E 6F 70  71 72 AA BA  E6 B8 C6 5D  # 90-9F
 B5 FC 73 74  75 76 77 78  79 7A A1 BF  D0 DD DE AE  # A0-AF
 A2 A3 A5 B7  A9 5B B6 BC  BD BE AC 7C  AF A8 B4 D7  # B0-BF
 E4 41 42 43  44 45 46 47  48 49 AD F4  A6 F2 F3 F5  # C0-CF
 E5 4A 4B 4C  4D 4E 4F 50  51 52 B9 FB  7E F9 FA FF  # D0-DF
 C9 F7 53 54  55 56 57 58  59 5A B2 D4  40 D2 D3 D5  # E0-EF
 30 31 32 33  34 35 36 37  38 39 B3 DB  DC D9 DA 9F  # F0-FF

# End
```

Spain (00819x00284) Code Page Translation Table

```
#================================================================
# 00819x00284
#      Character code page mapping tables for Spain (España).
#
# Local:    "IBM-CP00819", ISO-8859-1 Latin-1
# Remote:   "IBM-CP00284", EBCDIC Latin-1, Spain
# Built:    1999-04-16 19:50:00 UT
#
# @(#)$Id: 00819x00284,v 1.4.2.1 1999/04/29 13:04:22 cmadm Exp $
#-----------------------------------------------------------------
```

```
# Header
version 100;

# Outbound (local -> remote) table
table 256 1 1;
 00 01 02 03  37 2D 2E 2F  16 05 15 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  3C 3D 32 26  18 19 3F 27  1C 1D 1E 1F  # 10-1F
 40 BB 7F 69  5B 6C 50 7D  4D 5D 5C 4E  6B 60 4B 61  # 20-2F
 F0 F1 F2 F3  F4 F5 F6 F7  F8 F9 7A 5E  4C 7E 6E 6F  # 30-3F
 7C C1 C2 C3  C4 C5 C6 C7  C8 C9 D1 D2  D3 D4 D5 D6  # 40-4F
 D7 D8 D9 E2  E3 E4 E5 E6  E7 E8 E9 4A  E0 5A BA 6D  # 50-5F
 79 81 82 83  84 85 86 87  88 89 91 92  93 94 95 96  # 60-6F
 97 98 99 A2  A3 A4 A5 A6  A7 A8 A9 C0  4F D0 BD 07  # 70-7F
 04 06 08 09  0A 14 17 1A  1B 20 21 22  23 24 25 28  # 80-8F
 29 2A 2B 2C  30 31 33 34  35 36 38 39  3A 3B 3E FF  # 90-9F
 41 AA B0 B1  9F B2 49 B5  A1 B4 9A 8A  5F CA AF BC  # A0-AF
 90 8F EA FA  BE A0 B6 B3  9D DA 9B 8B  B7 B8 B9 AB  # B0-BF
 64 65 62 66  63 67 9E 68  74 71 72 73  78 75 76 77  # C0-CF
 AC 7B ED EE  EB EF EC BF  80 FD FE FB  FC AD AE 59  # D0-DF
 44 45 42 46  43 47 9C 48  54 51 52 53  58 55 56 57  # E0-EF
 8C 6A CD CE  CB CF CC E1  70 DD DE DB  DC 8D 8E DF  # F0-FF

# Inbound (remote -> local) table
table 256 1 1;
 00 01 02 03  80 09 81 7F  82 83 84 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  85 0A 08 86  18 19 87 88  1C 1D 1E 1F  # 10-1F
 89 8A 8B 8C  8D 8E 17 1B  8F 90 91 92  93 05 06 07  # 20-2F
 94 95 16 96  97 98 99 04  9A 9B 9C 9D  14 15 9E 1A  # 30-3F
 20 A0 E2 E4  E0 E1 E3 E5  E7 A6 5B 2E  3C 28 2B 7C  # 40-4F
 26 E9 EA EB  E8 ED EE EF  EC DF 5D 24  2A 29 3B AC  # 50-5F
 2D 2F C2 C4  C0 C1 C3 C5  C7 23 F1 2C  25 5F 3E 3F  # 60-6F
 F8 C9 CA CB  C8 CD CE CF  CC 60 3A D1  40 27 3D 22  # 70-7F
 D8 61 62 63  64 65 66 67  68 69 AB BB  F0 FD FE B1  # 80-8F
 B0 6A 6B 6C  6D 6E 6F 70  71 72 AA BA  E6 B8 C6 A4  # 90-9F
 B5 A8 73 74  75 76 77 78  79 7A A1 BF  D0 DD DE AE  # A0-AF
 A2 A3 A5 B7  A9 A7 B6 BC  BD BE 5E 21  AF 7E B4 D7  # B0-BF
 7B 41 42 43  44 45 46 47  48 49 AD F4  F6 F2 F3 F5  # C0-CF
 7D 4A 4B 4C  4D 4E 4F 50  51 52 B9 FB  FC F9 FA FF  # D0-DF
 5C F7 53 54  55 56 57 58  59 5A B2 D4  D6 D2 D3 D5  # E0-EF
 30 31 32 33  34 35 36 37  38 39 B3 DB  DC D9 DA 9F  # F0-FF

# End
```

Great Britain (00819x00285) Code Page Translation Table

```
#================================================================
# 00819x00285
#      Character code page mapping tables for Great Britain (UK).
#
# Local:    "IBM-CP00819", ISO-8859-1 Latin-1
# Remote:   "IBM-CP00285", EBCDIC Latin-1, UK
# Built:    1999-04-16 19:50:00 UT
#
# @(#)$Id: 00819x00285,v 1.5.2.1 1999/04/29 13:04:04 cmadm Exp $
#----------------------------------------------------------------

# Header
version 100;

# Outbound (local -> remote) table
table 256 1 1;
 00 01 02 03  37 2D 2E 2F  16 05 15 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  3C 3D 32 26  18 19 3F 27  1C 1D 1E 1F  # 10-1F
 40 5A 7F 7B  4A 6C 50 7D  4D 5D 5C 4E  6B 60 4B 61  # 20-2F
 F0 F1 F2 F3  F4 F5 F6 F7  F8 F9 7A 5E  4C 7E 6E 6F  # 30-3F
 7C C1 C2 C3  C4 C5 C6 C7  C8 C9 D1 D2  D3 D4 D5 D6  # 40-4F
 D7 D8 D9 E2  E3 E4 E5 E6  E7 E8 E9 B1  E0 BB BA 6D  # 50-5F
 79 81 82 83  84 85 86 87  88 89 91 92  93 94 95 96  # 60-6F
 97 98 99 A2  A3 A4 A5 A6  A7 A8 A9 C0  4F D0 BC 07  # 70-7F
 04 06 08 09  0A 14 17 1A  1B 20 21 22  23 24 25 28  # 80-8F
 29 2A 2B 2C  30 31 33 34  35 36 38 39  3A 3B 3E FF  # 90-9F
 41 AA B0 5B  9F B2 6A B5  BD B4 9A 8A  5F CA AF A1  # A0-AF
 90 8F EA FA  BE A0 B6 B3  9D DA 9B 8B  B7 B8 B9 AB  # B0-BF
 64 65 62 66  63 67 9E 68  74 71 72 73  78 75 76 77  # C0-CF
 AC 69 ED EE  EB EF EC BF  80 FD FE FB  FC AD AE 59  # D0-DF
 44 45 42 46  43 47 9C 48  54 51 52 53  58 55 56 57  # E0-EF
 8C 49 CD CE  CB CF CC E1  70 DD DE DB  DC 8D 8E DF  # F0-FF

# Inbound (remote -> local) table
table 256 1 1;
 00 01 02 03  80 09 81 7F  82 83 84 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  85 0A 08 86  18 19 87 88  1C 1D 1E 1F  # 10-1F
 89 8A 8B 8C  8D 8E 17 1B  8F 90 91 92  93 05 06 07  # 20-2F
 94 95 16 96  97 98 99 04  9A 9B 9C 9D  14 15 9E 1A  # 30-3F
 20 A0 E2 E4  E0 E1 E3 E5  E7 F1 24 2E  3C 28 2B 7C  # 40-4F
 26 E9 EA EB  E8 ED EE EF  EC DF 21 A3  2A 29 3B AC  # 50-5F
 2D 2F C2 C4  C0 C1 C3 C5  C7 D1 A6 2C  25 5F 3E 3F  # 60-6F
 F8 C9 CA CB  C8 CD CE CF  CC 60 3A 23  40 27 3D 22  # 70-7F
 D8 61 62 63  64 65 66 67  68 69 AB BB  F0 FD FE B1  # 80-8F
 B0 6A 6B 6C  6D 6E 6F 70  71 72 AA BA  E6 B8 C6 A4  # 90-9F
 B5 AF 73 74  75 76 77 78  79 7A A1 BF  D0 DD DE AE  # A0-AF
 A2 5B A5 B7  A9 A7 B6 BC  BD BE 5E 5D  7E A8 B4 D7  # B0-BF
 7B 41 42 43  44 45 46 47  48 49 AD F4  F6 F2 F3 F5  # C0-CF
 7D 4A 4B 4C  4D 4E 4F 50  51 52 B9 FB  FC F9 FA FF  # D0-DF
 5C F7 53 54  55 56 57 58  59 5A B2 D4  D6 D2 D3 D5  # E0-EF
 30 31 32 33  34 35 36 37  38 39 B3 DB  DC D9 DA 9F  # F0-FF

# End
```

France (00819x00297) Code Page Translation Table

```
#================================================================
# 00819x00297
#      Character code page mapping tables for France.
#
# Local:    "IBM-CP00819", ISO-8859-1 Latin-1
# Remote:   "IBM-CP00297", EBCDIC Latin-1, France
# Built:    1999-04-16 23:30:00 UT
#
# @(#)$Id: 00819x00297,v 1.4.2.1 1999/04/29 13:04:27 cmadm Exp $
#----------------------------------------------------------------

# Header
version 100;

# Outbound (local -> remote) table
table 256 1 1;
 00 01 02 03  37 2D 2E 2F  16 05 15 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  3C 3D 32 26  18 19 3F 27  1C 1D 1E 1F  # 10-1F
 40 4F 7F B1  5B 6C 50 7D  4D 5D 5C 4E  6B 60 4B 61  # 20-2F
 F0 F1 F2 F3  F4 F5 F6 F7  F8 F9 7A 5E  4C 7E 6E 6F  # 30-3F
 44 C1 C2 C3  C4 C5 C6 C7  C8 C9 D1 D2  D3 D4 D5 D6  # 40-4F
 D7 D8 D9 E2  E3 E4 E5 E6  E7 E8 E9 90  48 B5 5F 6D  # 50-5F
 A0 81 82 83  84 85 86 87  88 89 91 92  93 94 95 96  # 60-6F
 97 98 99 A2  A3 A4 A5 A6  A7 A8 A9 51  BB 54 BD 07  # 70-7F
 04 06 08 09  0A 14 17 1A  1B 20 21 22  23 24 25 28  # 80-8F
 29 2A 2B 2C  30 31 33 34  35 36 38 39  3A 3B 3E FF  # 90-9F
 41 AA B0 7B  9F B2 DD 5A  A1 B4 9A 8A  BA CA AF BC  # A0-AF
 4A 8F EA FA  BE 79 B6 B3  9D DA 9B 8B  B7 B8 B9 AB  # B0-BF
 64 65 62 66  63 67 9E 68  74 71 72 73  78 75 76 77  # C0-CF
 AC 69 ED EE  EB EF EC BF  80 FD FE FB  FC AD AE 59  # D0-DF
 7C 45 42 46  43 47 9C E0  D0 C0 52 53  58 55 56 57  # E0-EF
 8C 49 CD CE  CB CF CC E1  70 6A DE DB  DC 8D 8E DF  # F0-FF

# Inbound (remote -> local) table
table 256 1 1;
 00 01 02 03  80 09 81 7F  82 83 84 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  85 0A 08 86  18 19 87 88  1C 1D 1E 1F  # 10-1F
 89 8A 8B 8C  8D 8E 17 1B  8F 90 91 92  93 05 06 07  # 20-2F
 94 95 16 96  97 98 99 04  9A 9B 9C 9D  14 15 9E 1A  # 30-3F
 20 A0 E2 E4  40 E1 E3 E5  5C F1 B0 2E  3C 28 2B 21  # 40-4F
 26 7B EA EB  7D ED EE EF  EC DF A7 24  2A 29 3B 5E  # 50-5F
 2D 2F C2 C4  C0 C1 C3 C5  C7 D1 F9 2C  25 5F 3E 3F  # 60-6F
 F8 C9 CA CB  C8 CD CE CF  CC B5 3A A3  E0 27 3D 22  # 70-7F
 D8 61 62 63  64 65 66 67  68 69 AB BB  F0 FD FE B1  # 80-8F
 5B 6A 6B 6C  6D 6E 6F 70  71 72 AA BA  E6 B8 C6 A4  # 90-9F
 60 A8 73 74  75 76 77 78  79 7A A1 BF  D0 DD DE AE  # A0-AF
 A2 23 A5 B7  A9 5D B6 BC  BD BE AC 7C  AF 7E B4 D7  # B0-BF
 E9 41 42 43  44 45 46 47  48 49 AD F4  F6 F2 F3 F5  # C0-CF
 E8 4A 4B 4C  4D 4E 4F 50  51 52 B9 FB  FC A6 FA FF  # D0-DF
 E7 F7 53 54  55 56 57 58  59 5A B2 D4  D6 D2 D3 D5  # E0-EF
 30 31 32 33  34 35 36 37  38 39 B3 DB  DC D9 DA 9F  # F0-FF

# End
```

Belgium (00819x00500) Code Page Translation Table

```
#==============================================================
# 00819x00500
#       Character code page mapping tables for Belgium (Belgique).
#
# Local:    "IBM-CP00819", ISO-8859-1 Latin-1
# Remote:   "IBM-CP00500", EBCDIC Latin-1, Belgium
# Built:    1999-04-16 19:50:00 UT
#
# @(#)$Id: 00819x00500,v 1.4.2.1 1999/04/29 13:04:09 cmadm Exp $
#--------------------------------------------------------------
```

```
# Header
version 100;

# Outbound (local -> remote) table
table 256 1 1;
 00 01 02 03  37 2D 2E 2F  16 05 15 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  3C 3D 32 26  18 19 3F 27  1C 1D 1E 1F  # 10-1F
 40 4F 7F 7B  5B 6C 50 7D  4D 5D 5C 4E  6B 60 4B 61  # 20-2F
 F0 F1 F2 F3  F4 F5 F6 F7  F8 F9 7A 5E  4C 7E 6E 6F  # 30-3F
 7C C1 C2 C3  C4 C5 C6 C7  C8 C9 D1 D2  D3 D4 D5 D6  # 40-4F
 D7 D8 D9 E2  E3 E4 E5 E6  E7 E8 E9 4A  E0 5A 5F 6D  # 50-5F
 79 81 82 83  84 85 86 87  88 89 91 92  93 94 95 96  # 60-6F
 97 98 99 A2  A3 A4 A5 A6  A7 A8 A9 C0  BB D0 A1 07  # 70-7F
 04 06 08 09  0A 14 17 1A  1B 20 21 22  23 24 25 28  # 80-8F
 29 2A 2B 2C  30 31 33 34  35 36 38 39  3A 3B 3E FF  # 90-9F
 41 AA B0 B1  9F B2 6A B5  BD B4 9A 8A  BA CA AF BC  # A0-AF
 90 8F EA FA  BE A0 B6 B3  9D DA 9B 8B  B7 B8 B9 AB  # B0-BF
 64 65 62 66  63 67 9E 68  74 71 72 73  78 75 76 77  # C0-CF
 AC 69 ED EE  EB EF EC BF  80 FD FE FB  FC AD AE 59  # D0-DF
 44 45 42 46  43 47 9C 48  54 51 52 53  58 55 56 57  # E0-EF
 8C 49 CD CE  CB CF CC E1  70 DD DE DB  DC 8D 8E DF  # F0-FF

# Inbound (remote -> local) table
table 256 1 1;
 00 01 02 03  80 09 81 7F  82 83 84 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  85 0A 08 86  18 19 87 88  1C 1D 1E 1F  # 10-1F
 89 8A 8B 8C  8D 8E 17 1B  8F 90 91 92  93 05 06 07  # 20-2F
 94 95 16 96  97 98 99 04  9A 9B 9C 9D  14 15 9E 1A  # 30-3F
 20 A0 E2 E4  E0 E1 E3 E5  E7 F1 5B 2E  3C 28 2B 21  # 40-4F
 26 E9 EA EB  E8 ED EE EF  EC DF 5D 24  2A 29 3B 5E  # 50-5F
 2D 2F C2 C4  C0 C1 C3 C5  C7 D1 A6 2C  25 5F 3E 3F  # 60-6F
 F8 C9 CA CB  C8 CD CE CF  CC 60 3A 23  40 27 3D 22  # 70-7F
 D8 61 62 63  64 65 66 67  68 69 AB BB  F0 FD FE B1  # 80-8F
 B0 6A 6B 6C  6D 6E 6F 70  71 72 AA BA  E6 B8 C6 A4  # 90-9F
 B5 7E 73 74  75 76 77 78  79 7A A1 BF  D0 DD DE AE  # A0-AF
 A2 A3 A5 B7  A9 A7 B6 BC  BD BE AC 7C  AF A8 B4 D7  # B0-BF
 7B 41 42 43  44 45 46 47  48 49 AD F4  F6 F2 F3 F5  # C0-CF
 7D 4A 4B 4C  4D 4E 4F 50  51 52 B9 FB  FC F9 FA FF  # D0-DF
 5C F7 53 54  55 56 57 58  59 5A B2 D4  D6 D2 D3 D5  # E0-EF
 30 31 32 33  34 35 36 37  38 39 B3 DB  DC D9 DA 9F  # F0-FF

# End
```

Portugal (00819x00860) Code Page Translation Table

```
#=================================================================
# 00819x00860
#       Character code page mapping tables for Portugal.
#
# Local:     "IBM-CP00819", ISO-8859-1 Latin-1
# Remote:    "IBM-CP00860", ASCII IBM-PC graphics, Portugal
# Built:     1999-04-20 00:03:00 UT
#
# Caveats
#       The mapping between the two code pages is inexact, because
some
#       characters do not exist in both code sets.
#
# @(#)$Id: 00819x00860,v 1.4 1999/04/20 20:19:20 david Exp $
#-----------------------------------------------------------------

# Header
version 100;

# Outbound (local -> remote) table
table 256 1 1;
 00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  9E B0 16 17  18 19 1A 1B  1C 1D 1E 1F  # 10-1F
 20 21 22 23  24 25 26 27  28 29 2A 2B  2C 2D 2E 2F  # 20-2F
 30 31 32 33  34 35 36 37  38 39 3A 3B  3C 3D 3E 3F  # 30-3F
 40 41 42 43  44 45 46 47  48 49 4A 4B  4C 4D 4E 4F  # 40-4F
 50 51 52 53  54 55 56 57  58 59 5A 5B  5C 5D 5E 5F  # 50-5F
 60 61 62 63  64 65 66 67  68 69 6A 6B  6C 6D 6E 6F  # 60-6F
 70 71 72 73  74 75 76 77  78 79 7A 7B  7C 7D 7E 7F  # 70-7F
 B1 B2 B3 B4  B5 B6 B7 B8  B9 BA BB BC  BD BE BF C0  # 80-8F
 C1 C2 C3 C4  C5 C6 C7 C8  C9 CA CB CC  CD CE CF D0  # 90-9F
 FF AD 9B 9C  D1 D2 D3 15  D4 D5 A6 AE  AA D6 D7 D8  # A0-AF
 F8 F1 FD D9  DA E6 14 FA  DB DC A7 AF  AC AB DD A8  # B0-BF
 91 86 8F 8E  DE DF E0 80  92 90 89 E2  98 8B E3 E4  # C0-CF
 E5 A5 A9 9F  8C 99 E7 E9  E8 9D 96 EA  9A EC EE E1  # D0-DF
 85 A0 83 84  EF F0 F2 87  8A 82 88 F3  8D A1 F4 F5  # E0-EF
 EB A4 95 A2  93 94 F7 F6  ED 97 A3 F9  81 FB FC FE  # F0-FF

# Inbound (remote -> local) table
table 256 1 1;
 00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  B6 A7 16 17  18 19 1A 1B  1C 1D 1E 1F  # 10-1F
 20 21 22 23  24 25 26 27  28 29 2A 2B  2C 2D 2E 2F  # 20-2F
 30 31 32 33  34 35 36 37  38 39 3A 3B  3C 3D 3E 3F  # 30-3F
 40 41 42 43  44 45 46 47  48 49 4A 4B  4C 4D 4E 4F  # 40-4F
 50 51 52 53  54 55 56 57  58 59 5A 5B  5C 5D 5E 5F  # 50-5F
 60 61 62 63  64 65 66 67  68 69 6A 6B  6C 6D 6E 6F  # 60-6F
 70 71 72 73  74 75 76 77  78 79 7A 7B  7C 7D 7E 7F  # 70-7F
 C7 FC E9 E2  E3 E0 C1 E7  EA CA E8 CD  D4 EC C3 C2  # 80-8F
 C9 C0 C8 F4  F5 F2 DA F9  CC D5 DC A2  A3 D9 14 D3  # 90-9F
 E1 ED F3 FA  F1 D1 AA BA  BF D2 AC BD  BC A1 AB BB  # A0-AF
 15 80 81 82  83 84 85 86  87 88 89 8A  8B 8C 8D 8E  # B0-BF
 8F 90 91 92  93 94 95 96  97 98 99 9A  9B 9C 9D 9E  # C0-CF
 9F A4 A5 A6  A8 A9 AD AE  AF B3 B4 B8  B9 BE C4 C5  # D0-DF
 C6 DF CB CE  CF D0 B5 D6  D8 D7 DB F0  DD F8 DE E4  # E0-EF
 E5 B1 E6 EB  EE EF F7 F6  B0 FB B7 FD  FE B2 FF A0  # F0-FF

# End
```

Latin-1 – (00819x01047) Code Page Translation Table

```
#=================================================================
# 00819x01047
#      Character code page mapping tables.
#
# Local:    "IBM-CP00819", ISO-8859-1 Latin-1
# Remote:   "IBM-CP01047", EBCDIC Latin-1
# Built:    1999-04-22 23:40:00 UT
#
# @(#)$Id: 00819x01047,v 1.1.2.1 1999/04/29 13:04:13 cmadm Exp $
#-----------------------------------------------------------------

# Header
version 100;

# Outbound (local -> remote) table
table 256 1 1;
 00 01 02 03  37 2D 2E 2F  16 05 15 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  3C 3D 32 26  18 19 3F 27  1C 1D 1E 1F  # 10-1F
 40 5A 7F 7B  5B 6C 50 7D  4D 5D 5C 4E  6B 60 4B 61  # 20-2F
 F0 F1 F2 F3  F4 F5 F6 F7  F8 F9 7A 5E  4C 7E 6E 6F  # 30-3F
 7C C1 C2 C3  C4 C5 C6 C7  C8 C9 D1 D2  D3 D4 D5 D6  # 40-4F
 D7 D8 D9 E2  E3 E4 E5 E6  E7 E8 E9 AD  E0 BD 5F 6D  # 50-5F
 79 81 82 83  84 85 86 87  88 89 91 92  93 94 95 96  # 60-6F
 97 98 99 A2  A3 A4 A5 A6  A7 A8 A9 C0  4F D0 A1 07  # 70-7F
 04 06 08 09  0A 14 17 1A  1B 20 21 22  23 24 25 28  # 80-8F
 29 2A 2B 2C  30 31 33 34  35 36 38 39  3A 3B 3E FF  # 90-9F
 41 AA 4A B1  9F B2 6A B5  BB B4 9A 8A  B0 CA AF BC  # A0-AF
 90 8F EA FA  BE A0 B6 B3  9D DA 9B 8B  B7 B8 B9 AB  # B0-BF
 64 65 62 66  63 67 9E 68  74 71 72 73  78 75 76 77  # C0-CF
 AC 69 ED EE  EB EF EC BF  80 FD FE FB  FC BA AE 59  # D0-DF
 44 45 42 46  43 47 9C 48  54 51 52 53  58 55 56 57  # E0-EF
 8C 49 CD CE  CB CF CC E1  70 DD DE DB  DC 8D 8E DF  # F0-FF

# Inbound (remote -> local) table
table 256 1 1;
 00 01 02 03  80 09 81 7F  82 83 84 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  85 0A 08 86  18 19 87 88  1C 1D 1E 1F  # 10-1F
 89 8A 8B 8C  8D 8E 17 1B  8F 90 91 92  93 05 06 07  # 20-2F
 94 95 16 96  97 98 99 04  9A 9B 9C 9D  14 15 9E 1A  # 30-3F
 20 A0 E2 E4  E0 E1 E3 E5  E7 F1 A2 2E  3C 28 2B 7C  # 40-4F
 26 E9 EA EB  E8 ED EE EF  EC DF 21 24  2A 29 3B 5E  # 50-5F
 2D 2F C2 C4  C0 C1 C3 C5  C7 D1 A6 2C  25 5F 3E 3F  # 60-6F
 F8 C9 CA CB  C8 CD CE CF  CC 60 3A 23  40 27 3D 22  # 70-7F
 D8 61 62 63  64 65 66 67  68 69 AB BB  F0 FD FE B1  # 80-8F
 B0 6A 6B 6C  6D 6E 6F 70  71 72 AA BA  E6 B8 C6 A4  # 90-9F
 B5 7E 73 74  75 76 77 78  79 7A A1 BF  D0 5B DE AE  # A0-AF
 AC A3 A5 B7  A9 A7 B6 BC  BD BE DD A8  AF 5D B4 D7  # B0-BF
 7B 41 42 43  44 45 46 47  48 49 AD F4  F6 F2 F3 F5  # C0-CF
 7D 4A 4B 4C  4D 4E 4F 50  51 52 B9 FB  FC F9 FA FF  # D0-DF
 5C F7 53 54  55 56 57 58  59 5A B2 D4  D6 D2 D3 D5  # E0-EF
 30 31 32 33  34 35 36 37  38 39 B3 DB  DC D9 DA 9F  # F0-FF

# End
```

Latin-2 – (00912x00870) Code Page Translation Table

```
#===============================================================
# 00912x00870
#      Character code page mapping tables for Latin-2 character
sets.
#
# Local:    "IBM-CP00912", ISO-8859-2 Latin-2
# Remote:   "IBM-CP00870", EBCDIC Latin-2
# Built:    1999-04-16 19:50:00 UT
#
# @(#)$Id: 00912x00870,v 1.3.2.1 1999/04/29 13:04:32 cmadm Exp $
#----------------------------------------------------------------

# Header
version 100;

# Outbound (local -> remote) table
table 256 1 1;
 00 01 02 03  37 2D 2E 2F  16 05 15 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  3C 3D 32 26  18 19 3F 27  1C 1D 1E 1F  # 10-1F
 40 4F 7F 7B  5B 6C 50 7D  4D 5D 5C 4E  6B 60 4B 61  # 20-2F
 F0 F1 F2 F3  F4 F5 F6 F7  F8 F9 7A 5E  4C 7E 6E 6F  # 30-3F
 7C C1 C2 C3  C4 C5 C6 C7  C8 C9 D1 D2  D3 D4 D5 D6  # 40-4F
 D7 D8 D9 E2  E3 E4 E5 E6  E7 E8 E9 4A  E0 5A 5F 6D  # 50-5F
 79 81 82 83  84 85 86 87  88 89 91 92  93 94 95 96  # 60-6F
 97 98 99 A2  A3 A4 A5 A6  A7 A8 A9 C0  6A D0 A1 07  # 70-7F
 04 06 08 09  0A 14 17 1A  1B 20 21 22  23 24 25 28  # 80-8F
 29 2A 2B 2C  30 31 33 34  35 36 38 39  3A 3B 3E FF  # 90-9F
 41 B1 80 BA  9F 77 AA B5  BD BC AF FD  B9 CA B8 B4  # A0-AF
 90 A0 9E 9A  BE 57 8A 70  9D 9C 8F DD  B7 64 B6 B2  # B0-BF
 ED 65 62 66  63 78 69 68  67 71 72 73  DA 75 76 FA  # C0-CF
 AC BB AB EE  EB EF EC BF  AE 74 FE FB  FC AD B3 59  # D0-DF
 CD 45 42 46  43 58 49 48  47 51 52 53  DF 55 56 EA  # E0-EF
 8C 9B 8B CE  CB CF CC E1  8E 54 DE DB  DC 8D 44 B0  # F0-FF

# Inbound (remote -> local) table
table 256 1 1;
 00 01 02 03  80 09 81 7F  82 83 84 0B  0C 0D 0E 0F  # 00-0F
 10 11 12 13  85 0A 08 86  18 19 87 88  1C 1D 1E 1F  # 10-1F
 89 8A 8B 8C  8D 8E 17 1B  8F 90 91 92  93 05 06 07  # 20-2F
 94 95 16 96  97 98 99 04  9A 9B 9C 9D  14 15 9E 1A  # 30-3F
 20 A0 E2 E4  FE E1 E3 E8  E7 E6 5B 2E  3C 28 2B 21  # 40-4F
 26 E9 EA EB  F9 ED EE B5  E5 DF 5D 24  2A 29 3B 5E  # 50-5F
 2D 2F C2 C4  BD C1 C3 C8  C7 C6 7C 2C  25 5F 3E 3F  # 60-6F
 B7 C9 CA CB  D9 CD CE A5  C5 60 3A 23  40 27 3D 22  # 70-7F
 A2 61 62 63  64 65 66 67  68 69 B6 F2  F0 FD F8 BA  # 80-8F
 B0 6A 6B 6C  6D 6E 6F 70  71 72 B3 F1  B9 B8 B2 A4  # 90-9F
 B1 7E 73 74  75 76 77 78  79 7A A6 D2  D0 DD D8 AA  # A0-AF
 FF A1 BF DE  AF A7 BE BC  AE AC A3 D1  A9 A8 B4 D7  # B0-BF
 7B 41 42 43  44 45 46 47  48 49 AD F4  F6 E0 F3 F5  # C0-CF
 7D 4A 4B 4C  4D 4E 4F 50  51 52 CC FB  FC BB FA EC  # D0-DF
 5C F7 53 54  55 56 57 58  59 5A EF D4  D6 C0 D3 D5  # E0-EF
 30 31 32 33  34 35 36 37  38 39 CF DB  DC AB DA 9F  # F0-FF
# End
```

Glossary

A

Access Control Lists (ACL)

A Tuxedo security feature that controls client access to services by means of lists that are automatically checked each time a service is requested.

ACID Properties

The essential characteristic of transaction processing systems:
Atomicity: All changes that a transaction makes to a database are made permanent, or else are nullified.
Consistency: A successful transaction transforms a database from a previous valid state to a new valid state.
Isolation: Changes that a transaction makes to a database are not visible to other operations until the transaction completes its work.
Durability: Changes that a transaction makes to a database survive future system or media failures.

application

A BEA TUXEDO System/T *application* is bounded by the environment described in a single TUXCONFIG file. In /Domain, a BEA TUXEDO System/T application can communicate with another application via a domain gateway group.

application domain

When used alone, the term *Domain* can mean a number of things. In order to avoid confusion, the term *application domain* is used to refer to a BEA TUXEDO application bounded by the configuration of a `tmconfig` file. This application domain can be restricted to a single platform, or shared memory (SHM) environment, or could be scaled across multiple machines in a multiple processor (MP) environment.

Application Programming Interface (API)

1) The verbs and environment that exist at the application level to support a particular system software product. 2) A set of code that enables a developer to initiate and complete client/server requests within an application. 3) A set of calling conventions that define how to invoke a service. A set of well-defined programming interfaces (entry points, calling parameters, and return values) by which one software program utilizes the services of another

Application Program-to-Program Communication (APPC)

An interface to LU6.2 services; provides a set of primitives to conduct conversations in LU6.2 sessions.

Application-Transaction Monitor Interface (ATMI)

The Application Programming Interface (API) to Tuxedo that includes transaction routines, message handling routines, service interface routines, and buffer management routines.

C

client

A program designed to request information from a server.

CNOS

CNOS are service programs implemented as part of an LU6.2. The CNOS programs negotiate session limits between the two communication LU's.

Common Programming Interface for Communications (CPI-C)

An interface to LU6.2 services. It is a simpler set of primitives than the APPC interface and is intended for use in program-to-program communications.

conversation

In this guide *conversation* has two meanings; the context determines which meaning is intended. In BEA TUXEDO System/T, conversation identifies a mode of communication between processes in which a connection is opened and stays open until brought down. Communication is achieved through sends and receives. This is distinguished from the request/response model in which communication is achieved through calls and replies. In SNA terms, a conversation uses a session as long as the conversation continues. In an SNA conversation, communication can be either the BEA TUXEDO System/T conversation or request/response model. Each SNA conversation is assigned a CONVID (Conversation ID) at the time it is initialized by the LU. SNA conversations can be *mapped* or *basic*:

conversation, Mapped

Conversations that allow programmers to send and receive buffers without having to worry about the sizes of underlying request units (RUs) used for communication. The LU takes the buffer and divides it, if necessary, into appropriate Logical Records with associated length fields and data type fields. This is the style supported for applications by BEA eLink Adapter for Mainframe software.

conversation, basic

Conversations in which logical records with appropriate type and length fields must be formatted for transmission and parsed on receipt. The service transaction programs in an LU use basic conversations to communicate.

Communications Resource Manager Application Programming Interface (CR-MAPI)

The proprietary interface between the two primary eAM components, the GWS-NAX and the CRM.

Customer Information Control System/Extended System Architecture (CICS/ESA)

An operating environment devised by IBM that provides a foundation upon which to write customer applications programs. Several facilities useful for programming are supplied by the CICS environment, including basic mapping services (BMS), transient data queues (TD), temporary storage files (TS), memory services, etc. Customer applications are built as separate transaction programs, and are invoked as transactional tasks. CICS/ESA is a trademark of International Business Machines (IBM), Inc.

D

Distributed Program Link (DPL)

Function of CICS ISC that supports LINK requests between CICS regions, and is similar to a BEA TUXEDO request/response.

Distributed Transaction Processing (DTP)

A CICS intercommunication in which processing is distributed among transactions that communicate synchronously over intersystem or inter-region links. It is roughly equivalent to BEA TUXEDO conversations.

domain

A *domain* can be another BEA TUXEDO System/T application that is independently administered, an application that is under the control of another transaction processing system, or an application in a remote CICS/ESA region. Domains can be local or remote.

domain gateway

A BEA TUXEDO System/T process that provides connectivity to remote BEA TUXEDO application environments, such as OSI, MVS/APPC, CICS/MVS, and IMS operating environments. BEA eLink for Mainframe OSI TP, BEA eLink Adapter for Mainframe, and BEA eLink TCP/IP are domain gateways.

domain gateway group

A *Domain Gateway Group* is a collection of domain gateway processes that provide communication services with other domains.

E

ESA

(ESA) Enterprise Systems Architecture is the conceptual structure and functional behavior of IBM’s latest range of mainframe computers. ESA/370 is the fourth step in an evolution of which the first three steps were System/360, System/370, and System/370 extended architecture (370-XA).

F

Field Manipulation Language (FML)

A set of C language functions for defining and manipulating storage structures called field buffers. Cooperating processes can send and receive data in fielded buffers.

FML Buffer

A buffer of self-describing data items accessed through the Field Manipulation Language API.

G

graphical administrative interface

A Tuxedo System component that enables an authorized user to configure and control an application through a Motif-based set of screens and icons.

I

inbound

A generic term referring to request message direction relative to the server, or response message direction relative to the client.

Information Management System (IMS)

A database manager used by CICS/ESA to allow access to data. IMS provides for the arrangement of data in an hierarchical structure and a common access approach in application programs that manipulate IMS databases.

InterSystem Communications (ISC)

Communication between separate systems by means of SNA networking facilities or by means of the application-to-application facilities. ISC links CICS systems to other systems, and may be used for communication between user applications, or to transparently execute CICS functions on a remote CICS system.

J

Job Control Language (JCL)

Control language used to describe a job and its requirements to an operating system.

L

local domain

A *Local Domain* is a part of an application (set or subset of services) that is available to other domains. A Local Domain is always represented by a Domain Gateway Group, and the terms are used interchangeably.

local service

A *Local Service* is a service of a local domain that is made available to remote domains through a Domain Gateway Group.

Logical Unit (LU)

In SNA, a port through which a user gains access to the services of a network. Also, see System Network Architecture (SNA).

LU6.2

LU6.2 is a particular SNA logical unit that identifies a specific set of services for program to program communication. Services include syncpoint, mapping of buffers into records, message confirmation, and security.

M

MODENAME

MODENAME is a configuration parameter that names a set of characteristics for a group of BEA eLink Adapter for Mainframe sessions. In the CICS region, the mode is defined in VTAM and referenced in CIC and the DMCONFIG file.

mirror task

CICS/ESA task that services incoming requests that specify a *mirror transaction* (CSMI, CSM1, CSM2, CSM3, CSM5, CPMI, CVMI, or a user-defined mirror transaction identifier).

mirror transaction

CICS/ESA transaction that recreates a request that is function shipped from one system to another, issues the request on the second system, and passes the acquired data back to the first system.

mirror transaction identifier support

BEA eLink Adapter for Mainframe feature which enables BEA TUXEDO clients to invoke host CICS/ESA programs and, conversely, CICS/ESA client programs to invoke BEA TUXEDO services. Based on the IBM CICS/ESA mirror transaction.

Multiple Virtual Storage (MVS)

An operating system for processing systems consisting of one or more mainframe processors.

O

outbound

A generic term referring to request message direction relative to the client, or response message direction relative to the server.

P

Partitioned Data Set (PDS).

A CICS/ESA data set in direct access storage that is divided into partitions called members. A member can contain a program or data. Program libraries are held in partitioned data sets.

R

re-entrant

The attribute of a program or routine that allows the same copy of the program or routine to be used concurrently by two or more tasks.

remote domain

A *Remote Domain* is a part of an application accessed through a Domain Gateway Group. The remote domain may be another BEA TUXEDO System/T application, an application running under another TP system, or a BEA eLink Adapter for Mainframe application.

remote service

A *Remote Service* is a service of a remote domain that is made available to the local application through a Domain Gateway Group.

Resource Definition Online (RDO)

The recommended method of defining resources to CICS/ESA. Resource definitions are created interactively by a CEDA transaction, or by the DFHCSDUP utility. Both methods store definition in the CICS/ESA system definition data set (CSD). At CICS initialization, CSD definitions are selectively installed as CICS system tables controlled by a user-supplied list of definitions. CEDA-defined resource definitions can be installed while CICS is active and used immediately.

S

server

A computer or program that is dedicated to providing information in response to external requests.

session

When two LU’s bind with each other, that is, when they have successfully negotiated how they will communicate, they are said to be in *session*. SNA has fixed limits on the number of sessions configurable for an LU type.

stack

Platform vendor-supplied software that provides connectivity to an SNA network.

synchronization Level (sync level)

The level of synchronization (0, 1, or 2) established for an APPC session between intercommunicating CICS/ESA transactions. Level 0 gives no synchronization support, level 1 allows the exchange of private synchronization requests, and level 2 gives full CICS/ESA synchronization support, with backout of all updates to recoverable resources if failure occurs.

System Definition Data Set (CSD)

A VSAM KSDS cluster that contains a resource definition record for every resource defined to CICS using resource definition online (RDO).

SNA Communication Resource Manager (SNACRM)

A process that provides all of the sync-level two logic for an SNA domain gateway and directly communicates with the PU2.1 server.

SYM_DEST_NAME

A symbolic name for a combination of Partner LUNAME, MODENAME and TPNAME that uniquely identifies the destination for a conversation start-up request.

System Network Architecture (SNA)

A seven-layer networking protocol. Each layer of the protocol has a set of associated data communication services. The services of the uppermost layer are embodied in a Logical Unit (LU). Each LU type defined in SNA has its own specific set of services available to an end user for communicating. The end user may be a terminal device, or an application program. The SNA structure enables the end user to operate independently, unaffected by the specific facilities used for information exchange.

T

transaction

1) A complete unit of work that transforms a database from one consistent state to another. In DTP, a transaction can include multiple units of work performed on one or more systems.

2) A logical construct through which applications perform work on shared resources (e.g., databases). The work done on behalf of the transaction conforms to the four ACID Properties: atomicity, consistency, isolation, and durability.

Transaction Processing (TP)

A form of immediate data processing in which user requests are entered directly to the terminal and on-line programs satisfy the requests; for example, by updating database files and displaying output messages.

Transmission Control Protocol/Internet Protocol (TCP/IP)

The standard that permits two connected computers to establish a reliable connection. TCP/IP ensures reliable data delivery with a method known as Positive Acknowledgment with Retransmission (PAR).

typed buffer

A buffer for message communication involving data of a specific data type.

V

Virtual Telecommunications Access Method (VTAM)

A set of programs that control communication across a network between terminals and application programs.

Index