# BEA Tuxedo™ Mainframe Adapter for TCP

## Gateway User Guide

Version 8.1
Document Revised: November 14, 2003
Part Number:

# Copyright



# Restricted Rights Legend



# Trademarks or Service Marks

# Contents

## About This Document



## Introducing BEA Tuxedo Mainframe Adapter for TCP Gateway

# Understanding How BEA TMA TCP Gateway Works

# Configuring BEA TMA TCP Gateway for Data Mapping

# Setting Up Security for BEA TMA TCP Gateway



# Configuring BEA TMA TCP Gateway

# Index

# About This Document

BEA Tuxedo Mainframe Adapter for TCP Gateway (hereafter referenced as TMA TCP Gateway) is a gateway connectivity product that enables application programs on BEA Tuxedo systems to perform various non-transactional tasks with application programs that reside on different mainframe platforms.

This document describes the TMA TCP Gateway component and gives instructions for using the tools for building TMA TCP Gateway applications.

This guide explains how to configure and administer TMA TCP Gateway and how TMA TCP Gateway fits into the BEA Tuxedo environment. In addition, this guide:

- Explains how TMA TCP Gateway processes service requests, those that originate locally and those that originate on remote systems
- Explains how TMA TCP Gateway affects BEA Tuxedo application programs
- Provides conceptual and procedural information that will help you configure and administer TMA TCP Gateway

The BEA Tuxedo Mainframe Adapter for TCP Gateway User Guide includes the following topics:

- "Introducing BEA Tuxedo Mainframe Adapter for TCP Gateway" introduces TMA TCP Gateway, explains how TMA TCP Gateway fits into the BEA Tuxedo environment, and lists hardware and software requirements.
- "Understanding How BEA TMA TCP Gateway Works" explains how to start TMA TCP Gateway, how it processes services requests, and how it is terminated. It also provides

information that helps programmers develop clients' programs and service routines that send data through TMA TCP Gateway.

- "Configuring BEA TMA TCP Gateway for Data Mapping" provides detailed information about configuring the TMA TCP Gateway to convert application data between Tuxedo data types and mainframe data types.
- "Setting Up Security for BEA TMA TCP Gateway" provides information to set up security features that allow a requester from Tuxedo to pass a user ID requirement through the OTMA or CICS server interfaces for verification through system security.
- "Configuring BEA TMA TCP Gateway" provides detailed information for configuring eLink TCP Tuxedo by updating the `UBBCONFIG` file, specifying parameters in the `GWICONFIG` file, and defining domain configurations in the `DMCONFIG` file. These tasks must be complete for eLink TCP Tuxedo to properly interact with BEA Tuxedo software.
- "Starting BEA TMA TCP Gateway" explains how to set environmental variables, test connectivity with remote regions, and invoke TMA TCP Gateway.
- "Error and Information Messages" describes messages and resolutions for the system.
- "Code Page Translation Tables" provides the code page tables that are distributed with the product and instructions for modifying these tables.

# What You Need to Know

This document is intended for system administrators who will configure and administer TMA TCP Gateway. In addition, programmers will find useful pointers for developing client programs and service routines that send data through TMA TCP Gateway.

# 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.bea.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 Tuxedo Mainframe Adapter for TCP 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 Tuxedo Mainframe Adapter for TCP 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/.

# Related Information

The Tuxedo Mainframe Adapter for TCP documentation consists of the following items:

- *BEA Tuxedo Mainframe Adapter for TCP Release Notes*
- *BEA Tuxedo Mainframe Adapter for TCP Installation Guide*
- *BEA Tuxedo Mainframe Adapter for TCP Gateway User Guide*
- *BEA Tuxedo Mainframe Adapter for TCP CICS User Guide*
- *BEA Tuxedo Mainframe Adapter for TCP IMS User Guide*

For information about other BEA products, refer to http://edocs.bea.com/.

# Contact Us!

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

In your e-mail message, please indicate that you are using the documentation for the BEA Tuxedo Mainframe Adapter for TCP 8.1 release.

If you have any questions about this version of BEA Tuxedo Mainframe Adapter for TCP, or if you have problems installing and running BEA Tuxedo Mainframe Adapter for TCP, contact BEA Customer Support through BEA WebSupport at **www.bea.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

# 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. |
| `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.<br>*Examples*:<br>`#include <iostream.h> void main ( ) the pointer psz`<br>`chmod u+w *`<br>`\tux\data\ap`<br>`.doc`<br>`tux.doc`<br>`BITMAP`<br>`float` |
| **`monospace boldface text`** | Identifies significant words in code.<br>*Example*:<br>`void` **`commit`** `( )` |
| *`monospace italic text`* | Identifies variables in code.<br>*Example*:<br>`String` *`expr`* |

| Convention | Item |
|---|---|
| UPPERCASE TEXT | Indicates device names, environment variables, and logical operators.<br>*Example*s:<br>LPT1<br>SIGNON<br>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.<br>*Example*:<br>`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. |
| . . . | Indicates one of the following in a command line:<br>• That an argument can be repeated several times in a command line<br>• That the statement omits additional optional arguments<br>• That you can enter additional parameters, values, or other information<br>The ellipsis itself should never be typed.<br>*Example*:<br>`buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...` |
| .<br>.<br>. | Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed. |

CHAPTER 1

# Introducing BEA Tuxedo Mainframe Adapter for TCP Gateway

The BEA Tuxedo Mainframe Adapter for TCP Gateway (hereafter referenced as TMA TCP Gateway) product is a domains-based gateway connectivity feature that allows application programs on BEA Tuxedo systems to perform non-transactional tasks with application programs in other OLTP systems that support TMA TCP Gateway gateways. These include:

- CICS on IBM MVS systems
- IMS/TM on IBM MVS systems

The TMA TCP Gateway gateway is designed to provide transparent access to services that reside outside a BEA Tuxedo region. In addition, TMA TCP Gateway can provide remote application programs with access to local services.

This document provides information about the following topics:

- BEA TMA TCP Gateway and the BEA Tuxedo Architecture
- BEA TMA TCP Functionality
- GWIDOMAIN Gateway Component
- How TMA TCP Gateway Affects BEA Tuxedo Application Programs
- How TMA TCP Gateway Affects BEA Tuxedo Administration

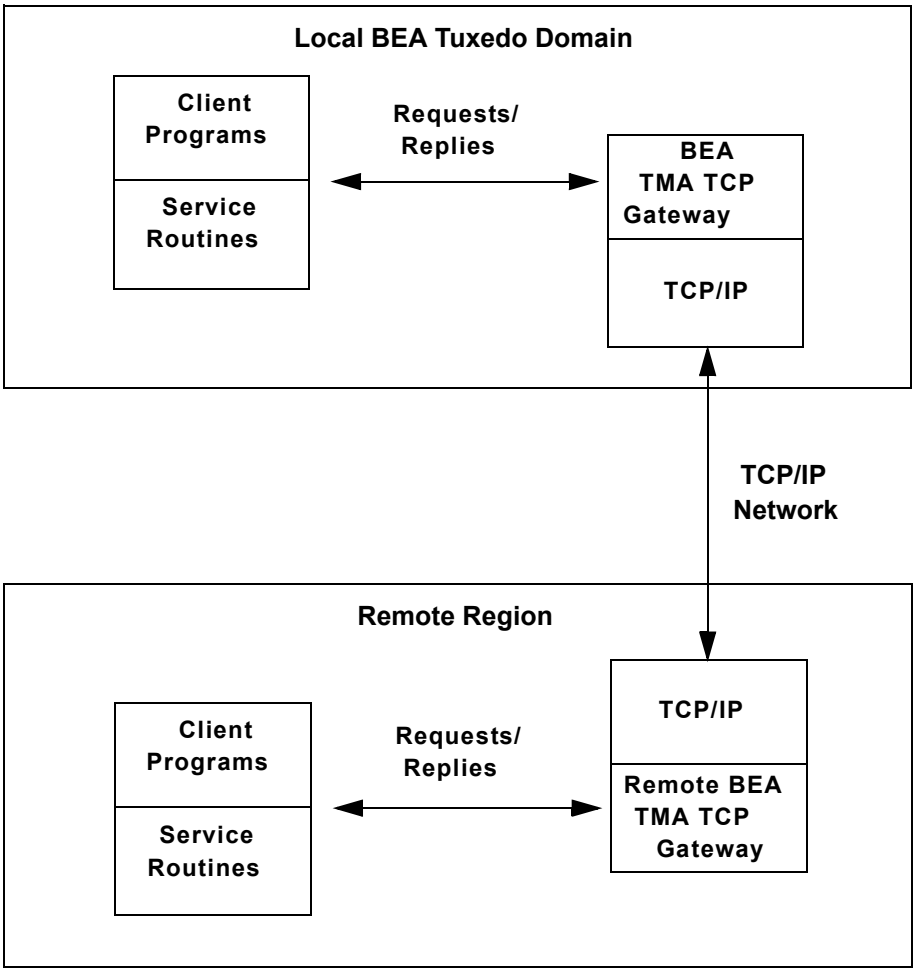
## BEA TMA TCP Gateway and the BEA Tuxedo Architecture

A BEA Tuxedo region consists of client and server programs that operate across a network of BEA Tuxedo systems or compatible systems. Any client program can request services that are

offered by any server program running on any computer in the region. The location of server programs is kept transparent through use of a directory that maps services to servers.

As Figure 1-1 shows, TMA TCP Gateway extends this transparent access by sending requests to and receiving requests from remote regions and systems through TCP/IP network software.

**Figure 1-1 Routing Service Calls through BEA TMA TCP Gateway**




As Figure 1-1 illustrates, inside a single region, TMA TCP Gateway fits between the BEA Tuxedo software and TCP/IP.

- When local client programs send requests to remote systems, TMA TCP Gateway transforms those requests into messages formatted appropriately for transmission to the remote system. Also, when remote systems respond, TMA TCP Gateway transforms these responses into replies that local client programs can process.
- When remote client programs send request messages, TMA TCP Gateway transforms those messages into requests that local service routines can process. Also, when local service routines send replies, TMA TCP Gateway transforms those replies into messages that remote services can process.

The TMA TCP Gateway product is implemented as a Tuxedo domain gateway. It accepts standard BEA Tuxedo service requests and returns standard replies.

One TMA TCP Gateway gateway connects to multiple communications targets, also referred to as gateways. Each communications target, or gateway, is a unique network endpoint.

Although remote systems are identified in the TMA TCP Gateway configuration, they remain unknown to BEA Tuxedo software. For example, remote systems that are accessible through TMA TCP Gateway are not identified in the `MACHINES` section of the `UBBCONFIG` file.

The TMA TCP Gateway gateway maintains its own control information in shared memory, in much the same way that BEA Tuxedo software itself maintains the Bulletin Board. Although TMA TCP Gateway accesses the BEA Tuxedo Bulletin Board, BEA Tuxedo does not access TMA TCP Gateway control information.

# Operational Considerations

Operational considerations are permanent limitations on using a feature. The following operational considerations apply to TMA TCP Gateway:

**Note:** In the following discussion, a local application program is one that resides within the immediate BEA Tuxedo administrative region. A remote application program is one that resides outside the immediate BEA Tuxedo administrative region.

- The TMA TCP Gateway software does not support conversational communication.
- The TMA TCP Gateway software supports only nontransactional communication.
- Local client and server programs which use the `tpsprio()` function set the priority where service requests are dequeued by TMA TCP Gateway. This process does not affect any prioritization on the remote system.
- Local client programs cannot use the `tpbroadcast()` function to send unsolicited messages to remote client programs (and the reverse).

- Local services cannot use the `tpbroadcast()` or `tpnotify()` functions to send messages to remote client programs (and the reverse).
- When local client and server programs use the `tpgprio()` function to determine the priority of a remote service, the priority of a local TMA TCP Gateway requester is returned.

For background information about these operational considerations, see "Understanding How BEA TMA TCP Gateway Works".

# BEA TMA TCP Functionality

The following functionality is available in this version of TMA TCP.

## Domains-based Gateway Connectivity

The TMA TCP product has a domains-based architecture supporting bidirectional communications, request/response support, and support for MVS Open Transaction Manager Access (OTMA) interfaces.

## Security

The TMA TCP Gateway product grants access to BEA Tuxedo services based on a user name that the remote gateway supplies.

The TMA TCP for CICS product can initiate transactions or link to programs. BEA Tuxedo security provides the user ID value to the TMA TCP product to test for appropriate security prior to initiating the transactions.

The TMA TCP for IMS product has an OTMA interface that supports enhanced security. This interface allows a BEA Tuxedo requester to pass a user ID through the OTMA server interface for authorization through a third-party security package, such as RACF.

## Connection Multiplexing

The TMA TCP Gateway allows multiple requests to process simultaneously over a single connection. This feature is known as connection multiplexing. Two connecting gateways determine a multiplex count that is acceptable to both sides at connection time. After establishing the connection, clients can send multiple requests (up to the number in the multiplex count) to the server gateway. Connection multiplexing allows for more efficient use of sockets and other system resources by the TMA TCP gateways.

**Note:** Each connection is one-directional, which means clients on opposing platforms cannot use the same connection to communicate with remote servers.

### Domain Name Server Support

The TMA TCP product supports domain name server (DNS) resolution of IP addresses. This support allows you to change the IP address at the Domain Name Server to implement address changes without reconfiguring the TMA TCP gateway.

## GWIDOMAIN Gateway Component

The TMA TCP product consists of a single component, the `GWIDOMAIN` gateway. This gateway is responsible for the mediating both incoming and outgoing requests. It also maintains connections with all remote gateways.

## How TMA TCP Gateway Affects BEA Tuxedo Application Programs

The TMA TCP Gateway product preserves the high degree of location transparency that BEA Tuxedo software provides. In fact, in virtually all cases, programmers do not need to know that particular services are provided by remote systems.

The TMA TCP Gateway product supports the main BEA Tuxedo communication paradigm: request/reply communications (either synchronous or asynchronous).

All BEA Tuxedo buffer types can be employed for data exchange. These include:

- X/Open standard XATMI buffer types
  - `X_OCTET`
  - `X_C_TYPE`
  - `X_COMMON`
- BEA Tuxedo ATMI buffers
  - `CARRAY`
  - `STRING`
  - `FML`
  - `VIEW`

Each of the three X/Open buffer types is equivalent to a BEA Tuxedo ATMI buffer type. The following information provides these equivalencies.

- X_OCTET is equivalent to CARRAY
- X_C_TYPE is equivalent to VIEW
- X_COMMON is equivalent to VIEW, but represents only the subset of field types that are common to both the C and COBOL languages

# VIEW Definitions

In some circumstances, you must convert typed buffers to formats that are acceptable to target systems. The standard BEA Tuxedo system VIEW definition mechanism is employed for this purpose.

VIEW definitions make it possible to map input data and output data between different programming environments (such as C and COBOL). They also enable TMA TCP Gateway to convert data representations automatically between different systems.

VIEW definitions can be created by programmers or system administrators. See the "Configuring BEA TMA TCP Gateway for Data Mapping" section for details. For more detailed information about programming considerations, see the "Understanding How BEA TMA TCP Gateway Works" section.

# FML Buffer Support

When communicating with systems or regions that do not support FML buffers directly, the TMA TCP Gateway can convert FML buffers to or from user-defined record layouts in a manner transparent to the FML application. Thus, once a VIEW definition that describes the remote application's record layout is created, it can be used to convert the record to or from an FML buffer. The GWICONFIG (TMA TCP Gateway configuration file) and DMCONFIG files contain VIEW specifications as part of the service description.

Through this conversion between ATMI buffers and record structures, TMA TCP Gateway supports sending fielded buffers containing FML data between regions. The TMA TCP Gateway software converts the data from FML buffers to user-defined records using the VIEW definitions and field descriptions at the originating region.

You can use an alternate data mapping tool to map FML buffers to formats that mainframe applications can use. For more information about how to configure TMA TCP Gateway to work

with an alternate data mapping tool, see the "Configuring BEA TMA TCP Gateway for Data Mapping" and "Configuring BEA TMA TCP Gateway" sections.

# How TMA TCP Gateway Affects BEA Tuxedo Administration

The TMA TCP Gateway administration tools and features are thoroughly integrated with BEA Tuxedo administration tools and features. Here are some specific examples:

- System administrators define TMA TCP Gateway in the BEA Tuxedo configuration as a regular Tuxedo domain gateway.
- The TMA TCP Gateway domain configuration file (`DMCONFIG`) specifies how local BEA Tuxedo service names are mapped to remote service names. Also, the `GWICONFIG` file identifies `VIEW` definitions that TMA TCP Gateway uses to convert and translate input and output data.
- At runtime, system administrators use BEA Tuxedo subcommands to manage TMA TCP Gateway and related processes.

For more detailed information about configuring TMA TCP Gateway, see the "Configuring BEA TMA TCP Gateway" section. For detailed information about commands for administering TMA TCP Gateway,see the *BEA Tuxedo Administrator's Guide* and the *BEA Tuxedo Domain User Guide*.

CHAPTER 2

# Understanding How BEA TMA TCP Gateway Works

To understand how Tuxedo Mainframe Adapter for TCP Gateway (hereafter referenced as TMA TCP Gateway) works, you need to understand how it performs the following tasks:

- Planning Your Configuration
- Initializing TMA TCP Gateway
- Processing Local Service Requests
- Processing Remote Service Requests
- Processing Shut Down Requests
- Programming Considerations

Each of these operations is described in the following subsections.

## Planning Your Configuration

One of the major benefits of using Tuxedo Mainframe Adapter for TCP Gateway to connect dissimilar systems is the degree to which different programming environments can be isolated. BEA Tuxedo programmers rarely need to know that services are handled by dissimilar systems or by systems in remote regions. Application programs do not need to be developed in any special way.

The key to this high degree of transparency is the TMA TCP Gateway configuration. Through TMA TCP Gateway configuration, environmental differences, such as naming conventions and data formats, are concealed from programmers and programs.

Three kinds of environmental differences are isolated in the TMA TCP Gateway configuration files (GWICONFIG and DMCONFIG). They are:

- Service names

  Different systems have different rules for naming services. Service names can differ in length, allowable characters, and even conventions as to how they are constructed or chosen.

- Input and output data formats

  Different systems have different conventions for formatting input and output data (such as structure, character set, and so forth).

- Error handling

  Different systems report application errors in different ways.

The technique that hides these differences is called mapping. Generally, when you map things, you associate local values or entities with values or entities that are meaningful to programs on remote systems.

The procedure for mapping service names is self-explanatory; you create a configuration file record in which a local name for a service is paired with a remote name for that service. On the other hand, procedures for mapping input data, output data, and application errors are more complex. Conceptual information and other background information are required.

For detailed information about updating the TMA TCP Gateway configuration files (GWICONFIG and DMCONFIG), see the "Configuring BEA TMA TCP Gateway" section.

**Note:** All TMA TCP Gateway configuration parameters are described in the "Configuring BEA TMA TCP Gateway" section. This document focuses on complex parameters that require a separate introduction.

The task of configuring data mappings could be considered a programming activity because it requires knowledge of the BEA Tuxedo programming environment. However, because configuration parameters affect many application programs, configuration is usually an administrator's responsibility.

# Initializing TMA TCP Gateway

When you boot BEA Tuxedo software using the tmboot command, TMA TCP Gateway initializes in the following manner.

1. The TMA TCP Gateway software parses the GWICONFIG configuration file and initializes all parameters. If syntax errors are encountered during parsing, TMA TCP Gateway writes a message to the ULOG file and initialization fails.
2. After reading the GWICONFIG file, TMA TCP Gateway advertises remote services that are named in the file dynamically. These services includes services for all remote gateways.

# Processing Local Service Requests

When TMA TCP Gateway receives a BEA Tuxedo service request from a local client program, it processes the request in the following manner.

## Step 1: Receiving a Service Request from BEA Tuxedo Software

When a client program sends a request for a remote service that is accessible through TMA TCP Gateway, BEA Tuxedo forwards the request to the gateway pending the requested service.

## Step 2: Connecting to a Remote System

The TMA TCP Gateway will determine which remote system will process each request. Data-dependent routing rules may be used to determine the desired remote system.

If no connection to the target remote system exists or an existing connection has been broken, the TMA TCP Gateway opens a new connection at this time.

If the remote system returns a connection failure indication, the BEA Tuxedo service request fails and an error is returned to the caller. The actual error value returned depends on the timing of the connection failure. Information about failures is written to the ULOG file.

## Step 3: Converting Input Buffer Types

In some circumstances, typed buffers associated with service requests must be converted before service requests can be sent to remote systems. Type conversion involves changing the layout of a buffer to a format that is acceptable to a remote service.

For example, if a local client program places user input in an FML buffer that a remote service cannot process, the buffer must be converted into the structure the remote service expects.

In situations where input type conversion is required, programmers or administrators must perform the following tasks:

- Determine the format of the input data the remote service expects.

- If necessary, create a VIEW definition that describes the format of the input data. VIEW definitions are descriptions of data structures that are used for input and output in the BEA Tuxedo environment.
- Specify this information in the TMA TCP Gateway configuration files (GWICONFIG and DMCONFIG).

Once these tasks have been completed, TMA TCP Gateway performs all necessary type conversions automatically.

For information about creating VIEW definitions to facilitate type conversion, see "Configuring BEA TMA TCP Gateway for Data Mapping." For information about the TMA TCP Gateway configuration file, see "Configuring BEA TMA TCP Gateway."

## Step 4: Translating Input Data

The TMA TCP Gateway software automatically translates data as required. Translation refers to a change in how intrinsic data types are represented with respect to word length, byte ordering, and character encoding.

To facilitate data translation, administrators must specify certain parameters in the GWICONFIG configuration file. For detailed information about how TMA TCP Gateway translates data, refer to the "Configuring BEA TMA TCP Gateway for Data Mapping" section.

## Step 5: Transmitting the Service Request

The TMA TCP Gateway product constructs a request message. This message includes the following items and is sent to the remote system:

- The remote service name
- The input data record that TMA TCP Gateway has converted and translated, as required
- An indication of whether the remote service should return a reply to the caller

## Step 6: Receiving a Reply

After sending a request message, TMA TCP Gateway performs a receive operation. If the TMA TCP Gateway receive timeout expires before a message arrives from the remote system, a TPETIME error is returned to the caller.

## Step 7: Translating the Reply

After TMA TCP Gateway receives a reply, data representations are translated as needed, in the reverse of the input translation. For details, see the "Step 4: Translating Input Data" section in this document.

## Step 8: Converting Output Data

If the format of the reply is not suitable for the local client program, TMA TCP Gateway converts the reply into an appropriate buffer format.

In situations where output conversion is required, programmers or administrators must do the following:

- Determine the type and the format of the output buffer the local client program expects
- If necessary, create a `VIEW` definition that describes the format of the output buffer
- Specify this information in the TMA TCP Gateway configuration files (`GWICONFIG` and `DMCONFIG`)

Once these tasks have been completed, TMA TCP Gateway performs all necessary conversions automatically.

For information about creating `VIEW` definitions to facilitate type conversion, see the "Configuring BEA TMA TCP Gateway for Data Mapping" section. For information about the TMA TCP Gateway configuration file, see the "Configuring BEA TMA TCP Gateway" section.

## Step 9: Sending the Reply to the Caller

The BEA Tuxedo buffer resulting from output type conversion and output data translation is returned to the caller with a `TPSUCCESS` or `TPFAIL` indication.

# Processing Remote Service Requests

The TMA TCP Gateway software processes remote service requests (those which originate on remote systems) in much the same way that it processes local requests. The following list offers a brief summary.

1. The TMA TCP Gateway receives the service request and, if necessary, the input record associated with the request is translated and converted into the specified input buffer format.
2. Acting as a client program, the gateway passes the service request to BEA Tuxedo software.

3. When the service routine is complete, the output buffer is converted into the specified output record format and/or translated, when required.
4. Finally, the output is sent to the requester on the remote system.

For more detailed information about record and buffer conversion, and data translation, see the "Processing Local Service Requests" section.

# Processing Shut Down Requests

When you send a shutdown request to TMA TCP Gateway using the `tmshutdown` command, TMA TCP Gateway performs the following tasks.

1. Completes outstanding requests
2. Drops all open connections
3. Terminates

# Programming Considerations

In general, BEA Tuxedo application programs that send requests through TMA TCP Gateway are developed the same as other BEA Tuxedo application programs.

The TMA TCP Gateway product supports all request/reply communications functions that are included in the ATMI and XATMI interfaces. In addition, all supported functions can be used in the standard manner documented in the BEA Tuxedo documentation.

- All XATMI and ATMI buffer types are supported.
- All system errors are reported in the standard manner as documented in the BEA Tuxedo documentation.

## Input and Output Issues

This subsection describes several input and output issues that programmers need to consider when developing application programs that use TMA TCP Gateway.

### Preparing Input and Output Data with TMA TCP Gateway

The "Processing Local Service Requests" section describes many circumstances that require conversion of input and output parameters into formats acceptable to remote systems or regions and the local system.

The TMA TCP Gateway product provides powerful configuration capabilities that make it possible for you to convert or map parameters easily rather than requiring you to program in a different way.

The TMA TCP Gateway configuration files (GWICONFIG and DMCONFIG) are a centralized mechanism that you can use to define and maintain relationships between the local system and remote systems or regions. In addition to input and output parameter mappings, these relationships include service name mappings (where remote service names are mapped to local service names) and error record mappings.

For more information about the GWICONFIG configuration file, see the "Configuring BEA TMA TCP Gateway" section.

## Service Request Parameters

BEA Tuxedo application programs can request the following two categories of remote services through TMA TCP Gateway:

- Existing application programs that were originally developed for traditional OLTP environments and have been adapted for use with BEA Tuxedo.
- Services that were developed specifically for the XATMI or ATMI (BEA Tuxedo) environments. Included in this category are services that reside in remote BEA Tuxedo regions.

If a remote service was developed specifically for the BEA Tuxedo environment, the input it requires is shaped by three factors:

- Application-specific requirements
- Normal ATMI or XATMI requirements for defining and sending input
- Machine-specific requirements for how input is formatted (such as those described in "Processing Local Service Requests")

On the other hand, if a remote service is an existing OLTP application program, additional requirements for input are often required. For example, many systems require input that includes terminal data.

Often, by creative use of the TMA TCP Gateway configuration capabilities previously introduced, you can eliminate the need to include control information, such as terminal data, in the BEA Tuxedo application source code that you develop. For instance, you can include terminal control codes in VIEW definitions that are associated with the TMA TCP Gateway configuration.

For information about the normal input requirements of BEA Tuxedo services, see the BEA Tuxedo *Programmer's Guide.*

## Output Data Considerations

To maintain the location transparency of the BEA Tuxedo environment, TMA TCP Gateway does not preserve data from BEA Tuxedo input buffers in the associated output buffers. Hence, the consequences of using the same buffer for input and output must be understood to avoid problems.

In particular, some existing BEA Tuxedo applications may use FML buffers to accumulate results or to maintain application context across service requests. Developers adding TMA TCP Gateway to such an application must do one of the following:

- Maintain a copy of the necessary data in the client program (or service routine) that makes requests.
- Ensure that the remote service returns input data with the output record.

This requirement is no different from the requirement that existed before the use of TMA TCP Gateway. That is, application programs that accumulate output data in FML buffers must ensure that services return replies in original FML input buffers (with output data added)—not in new or re-initialized buffers.

# Limitations on the Use of Certain ATMI Functions

The ATMI interface includes several features and functions—related primarily to conversations, transactions and client identities—that application programs cannot propagate to other application programs through TMA TCP Gateway.

In this guide, permanent limitations of this sort are referred to as operational considerations. Specific operational considerations are described in the following sections.

**Note:** In this discussion, a local application program is one that resides within the immediate BEA Tuxedo region. A remote application program is one that resides outside the immediate BEA Tuxedo region.

## Conversational Communication Functions

Conversational communication functions are subject to the following operational considerations:

- Local client and server programs cannot use the `tpconnect()` function to establish conversations with remote services.

- Remote client and server programs cannot use the `tpconnect()` function to establish conversations with local services.
- Similarly, the `tpsend()`, `tprecv()`, and `tpdiscon()` functions may not be used for communication through TMA TCP Gateway.

## Non-Transactional Communications

The TMA TCP Gateway product supports only non-transactional communications. Therefore, all communications via TMA TCP Gateway are subject to the following operational considerations:

- Remote services called by local client or server programs are always invoked outside the boundaries of any local transaction.
- Local services called by remote client or server programs are always invoked outside the boundaries of any local transaction.
- Local client or server programs calling remote services should invoke the `tpcall()` function, or the `tpacall()` and `tpgetrply()` functions, outside the boundaries of any local transaction, such as `tx_begin()`/`tx_commit()` or `tpbegin()`/`tpcommit()` pair.
- If local client or server programs must call remote services within the boundaries of a local transaction, `TPNOTRAN` must be specified as one of the flags to the `tpcall()` or `tpacall()` function.

## The tpsprio( ) and tpgprio( ) Functions

The `tpsprio()` and `tpgprio()` functions are subject to the following operational considerations:

- Local client and server programs cannot use the `tpsprio()` function to set the priority at which remote services are processed. Instead, a call to the `tpsprio()` function causes the priority of a local TMA TCP Gateway to be set.
- Remote client and server programs cannot use the `tpsprio()` function to set the priority at which local services are processed.
- When local client or server programs use the `tpgprio()` function to determine the priority of a remote service, the priority of a local TMA TCP Gateway is returned.

## The tpbroadcast() and tpnotify() Functions

The `tpbroadcast()` and `tpnotify()` functions are subject to the following operational considerations:

- Local client programs cannot use the `tpbroadcast()` function to send unsolicited messages to remote client programs (and the reverse).
- Local services cannot use the `tpbroadcast()` or `tpnotify()` functions to send messages to remote client programs (and the reverse).

# Error Handling

Three kinds of errors can be encountered by local application programs when they send requests through TMA TCP Gateways:

- Gateway errors
- Problems on remote systems
- Errors from remote application programs

The following sections explain how TMA TCP Gateway handles these different kinds of errors.

## Gateway Errors

When local or remote gateway errors occur, they are logged in the BEA Tuxedo `ULOG` file and associated service requests fail. Also, appropriate error codes are returned to callers.

## Remote System Failures

When remote systems encounter problems, service requests may fail or time out. The exact outcome depends on whether the remote system provides a means for TMA TCP Gateway to detect failure.

If the remote target system does not make it possible for TMA TCP Gateway to detect particular types of failure, the TMA TCP Gateway blocking timeout parameter can be tuned to provide timely detection of the problem.

For more information about the blocking timeout parameter, see the "Configuring BEA TMA TCP Gateway" section.

## Application Errors

Application errors are similar to remote system failures. Remote systems may or may not use error indicators to pass information back to the local TMA TCP Gateway resulting in the generation of error messages. If no such error indicators exist, service routines typically use their own mechanisms to report failures to callers.

When application errors occur, some service routines may not return their usual output records at all. Instead, they may return some other data indicating that there has been an error, such as a string that contains a failure message.

When TMA TCP Gateway receives a service failure message from a remote system, it:

- Converts the output buffer as required based on instructions it finds in the `GWICONFIG` file
- Writes an error record in the `ULOG` file if the configuration directs it to do so
- Returns the appropriate type of BEA Tuxedo buffer to the caller

**Note:** When BEA Tuxedo applications detect service failures, they should not assume that returned buffers are the expected type. BEA Tuxedo application programs may ignore return buffers when failures occur. If you need to check a buffer type, use the BEA Tuxedo `tptypes()` function. When the type is known, the buffer can be handled accordingly (for example, by displaying a window containing an error string).

CHAPTER 3

# Configuring BEA TMA TCP Gateway for Data Mapping

The key to the high degree of transparency between systems is the Tuxedo Mainframe Adapter for TCP Gateway (hereafter referenced as TMA TCP Gateway) configuration. Environmental differences, such as data formats, are concealed from programmers and programs through this mechanism.

This document provides information about the following topics:

- Converting Input and Output Data
- Managing Parameters for Buffer and Record Conversion
- Translating Data
- Encoding COBOL Data Types
- Using Code Page Translation Tables

This document also provides information about creating `VIEW` definitions. `VIEW` definitions are descriptions of data structures that are used for input and output in the BEA Tuxedo environment. The TMA TCP Gateway product uses `VIEW` definitions to determine how to convert input data and output data into formats that are acceptable to target systems.

For detailed information about updating the TMA TCP Gateway configuration files (`GWICONFIG` and `DMCONFIG`), see the "Configuring BEA TMA TCP Gateway" section.

**Note:** The task of configuring data mappings could be considered a programming activity because it requires knowledge of the BEA Tuxedo programming environment. However, because configuration parameters affect many application programs, configuration is usually an administrator's responsibility.

# Converting Input and Output Data

This section introduces procedures that TMA TCP Gateway follows to process and convert input and output data.

## Buffers and Records

In this guide, the following terms are used to describe input and output data.

**Buffer**

Input or output data as it exists inside the local BEA Tuxedo region. This definition includes all the buffer types that BEA Tuxedo software supports—both BEA Tuxedo ATMI buffer types and X/Open XATMI buffer types.

**Record**

Input or output data as it exists outside the local BEA Tuxedo region—on different kinds of systems.

These terms make it easier to understand how TMA TCP Gateway handles input and output data.

## Buffers Received from Local Programs

The TMA TCP Gateway processes buffers from local programs in the following manner.

1. When TMA TCP Gateway receives a buffer from a local program, it automatically determines the buffer's type.

   The TMA TCP Gateway product automatically “types” input buffers that local client programs send to remote services.

   The TMA TCP Gateway product automatically “types” output buffers that local services return to remote client programs.

2. After TMA TCP Gateway determines a buffer’s type, it consults the configuration file (`GWICONFIG`) to determine whether the buffer needs to be converted to a different format.

   Client requests sent to remote services may need to be converted to record formats that are meaningful to those services.

   Server responses returned to remote client programs may need to be converted to record formats that are meaningful to those programs.

3. If the configuration indicates that conversion is required, TMA TCP Gateway transforms the buffer into the record format that is specified in the configuration.

## Records Received from Remote Programs

The TMA TCP Gateway processes buffers from remote programs in the following manner.

1. When TMA TCP Gateway receives a record from a remote system, it consults the configuration file (GWICONFIG) to determine the record's type.
2. After TMA TCP Gateway determines a record's type, it consults the domain configuration (DMCONFIG) to determine whether the record needs to be converted to a different format.

   Client requests from remote client programs may need to be converted to buffer formats that are acceptable to local service routines.

   Server responses returned from remote services may need to be converted to buffer formats that are acceptable to local client programs.
3. If the configuration indicates that conversion is required, TMA TCP Gateway transforms the record into the buffer format that is specified in the configuration.

# Managing Parameters for Buffer and Record Conversion

The TMA TCP Gateway product provides four configuration parameters you can use to map buffers and records. For more information about buffers and records, see the "Buffers and Records" section.

Specify the following buffer configuration parameters in the domain configuration file (DMCONFIG).

INBUFTYPE
: Identifies the type, and in some cases the format, of a buffer received from a Tuxedo client or server

OUTBUFTYPE
: Identifies the type, and in some cases the format, of a buffer to be sent to a Tuxedo client or server

Specify the following record configuration parameters in the gateway configuration file (GWICONFIG).

INRECTYPE
: Identifies the type, and in some cases the format, of a buffer to be sent to a remote gateway

OUTRECTYPE
: Identifies the type, and in some cases the format, of a buffer received from a remote gateway

Each of these four parameters has two possible meanings or interpretations—one for service requests that originate locally, and one for service requests that originate on remote systems.

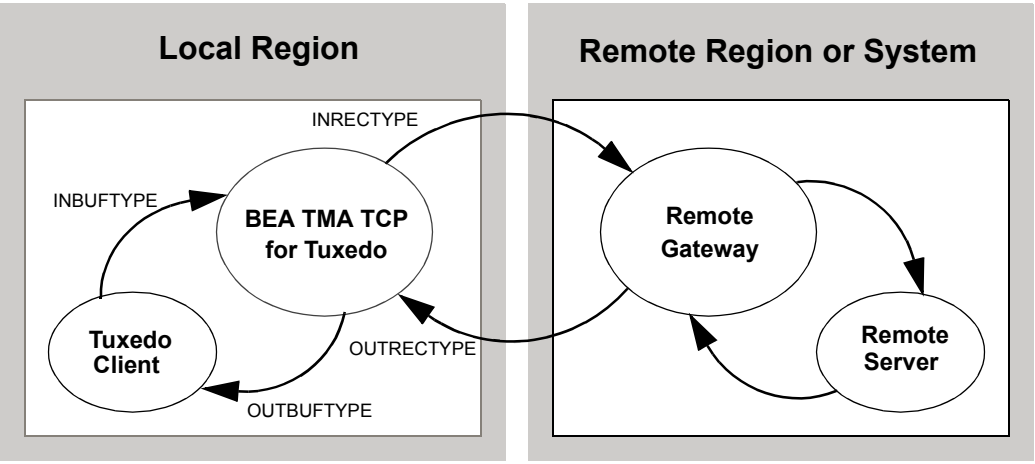
The following sections, "Parameters for Locally Originated Calls" and "Parameters for Remotely Originated Calls", explore these different meanings in detail.

# Parameters for Locally Originated Calls

This section takes a closer look at how TMA TCP Gateway handles service calls that originate locally, within the immediate BEA Tuxedo region. Also, it explains how the INBUFTYPE, INRECTYPE, OUTRECTYPE, and OUTBUFTYPE parameters can be used to manage the conversion of buffers and records that flow between local client programs and remote services.

In the following figure, a local BEA Tuxedo client program issues a service call that a local TMA TCP Gateway routes to a remote server through TMA TCP Gateway.

**Figure 3-1 How Parameters Are Mapped During Locally Originated Calls**




In this situation, the four configuration parameters that are shown in the figure have the following meanings:

- The INBUFTYPE parameter describes the BEA Tuxedo input buffer that the local client program provides to the TMA TCP Gateway through BEA Tuxedo software.
- The INRECTYPE parameter describes the input record that is sent to the service on the remote system.

- The OUTRECTYPE parameter describes the output record that is received from the service on the remote system.
- The OUTBUFTYPE parameter describes the BEA Tuxedo output buffer that is returned to the local client program.

## Guidelines for Mapping Input Buffers to Input Records

The following sections provide detailed information explaining how to use the INBUFTYPE and INRECTYPE parameters for service calls that originate locally (where local client programs call remote services).

INBUFTYPE

The INBUFTYPE parameter is used to specify the request buffer type that is provided to a local TMA TCP Gateway when a local client program issues a service request.

Because the gateway determines the type of client request buffers automatically at runtime, this parameter is described here for conceptual completeness only.

INRECTYPE

The INRECTYPE parameter is used to specify the type, and in some cases the format, of the request record that a particular remote service requires. The TMA TCP Gateway uses this information to convert BEA Tuxedo request buffers into records that remote services can process.

You must specify the INRECTYPE parameter when one of the circumstances described in the following table is true.

| Circumstance | Explanation |
|---|---|
| The remote service uses an input record that is structured differently than the client program's request buffer. | In this circumstance, the remote service uses a record that is structured differently than the client program's VIEW, X_C_TYPE, or X_COMMON buffer. For example, the remote service may expect structure members to be sequenced differently. |
| The remote service uses a request record that differs from the client program's request buffer in both type and structure. | In this case, the client program uses a BEA Tuxedo FML buffer and the remote service expects a corresponding record with an appropriate structure. |

The INRECTYPE parameter may be omitted if the request buffer is identical, in type and structure, to the request record the remote service expects.

## Guidelines for Mapping Output Records to Output Buffers

The following sections provide detailed information explaining how to use the OUTRECTYPE and OUTBUFTYPE parameters for service calls that originate locally (where local client programs call remote services and receive output from those services).

OUTBUFTYPE

The OUTBUFTYPE parameter is used to specify the type, and in some cases the structure, of the reply buffer that a local client program expects. The TMA TCP Gateway uses this information to map reply records from remote services to the appropriate kinds of reply buffers.

OUTRECTYPE

The OUTRECTYPE parameter is used to specify the type, and in some cases the format, of the reply record that a particular remote service returns to the local TMA TCP Gateway.

You must specify the OUTRECTYPE parameter when one of the circumstances described in the following table is true.

| Circumstance | Explanation |
|---|---|
| The remote service returns a reply record that is structured differently than the reply buffer the local client program expects. | In this circumstance, the remote service returns a record that is structured differently than the client program's VIEW, X_C_TYPE, or X_COMMON buffer. For example, the structure members of the output record may be sequenced differently than the structure members of the output buffer. |
| The remote service returns a reply record that differs in both type and structure from the reply buffer the client program expects. | In this case, the remote service returns a particular record and the local client program expects a corresponding BEA Tuxedo FML buffer. |

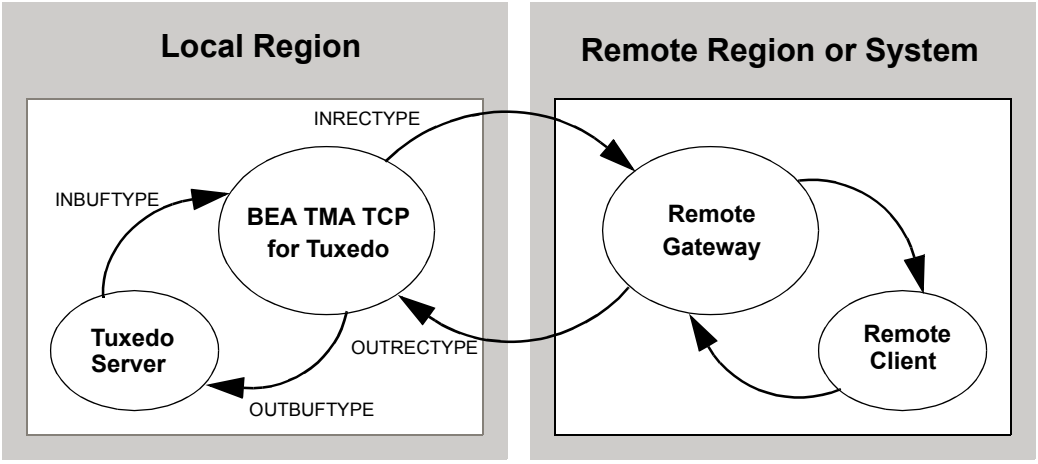
The OUTRECTYPE parameter may be omitted if the remote service returns a reply record that is identical, in type and structure, to the reply buffer the local client program expects.

# Parameters for Remotely Originated Calls

This section takes a closer look at how TMA TCP Gateway handles service calls that originate on remote computers, outside the local BEA Tuxedo region. Also, it explains how the INRECTYPE, INBUFTYPE, OUTBUFTYPE, and OUTRECTYPE parameters can be used to manage the conversion of buffers and records that flow between remote client programs and local services.

In the following figure, a remote client program issues a service request that a remote TMA TCP gateway routes to the local TMA TCP Gateway. The gateway receives the request from the network and passes the request to a local BEA Tuxedo server.

**Figure 3-2 How Parameters Are Mapped During Remotely Originated Calls**




In this situation, the four configuration parameters that are shown in the figure have the following meanings:

- The `OUTRECTYPE` parameter describes the output record that the remote client sends to the TMA TCP Gateway.
- The `OUTBUFTYPE` parameter describes the BEA Tuxedo output buffer that is provided to the local server.
- The `INBUFTYPE` parameter describes the BEA Tuxedo input buffer that the local server returns to the TMA TCP Gateway.
- The `INRECTYPE` parameter describes the input record that the local TMA TCP Gateway returns to the remote client program.

## Guidelines for Mapping Input Records to Input Buffers

The following sections provide detailed information explaining how to use the `INRECTYPE` and `INBUFTYPE` parameters for service calls that originate on remote systems (where remote client programs call local services).

INBUFTYPE

The INBUFTYPE parameter is used to specify the type, and in some cases the structure, of the reply buffer that the TMA TCP Gateway expects from a local server. The TMA TCP Gateway uses this information to map reply buffers from local server programs to the appropriate kind of reply records.

Because the gateway determines the type of incoming buffers automatically at runtime, this parameter is described here for conceptual completeness only.

INRECTYPE

The INRECTYPE parameter is used to specify the type, and in some cases the format, of the reply record that the local TMA TCP Gateway sends to the remote client.

You must specify the INRECTYPE parameter when one of the circumstances described in the following table is true.

| Circumstance | Explanation |
|---|---|
| The remote client program requires a reply record that is structured differently than the reply buffer the local service provides. | In this circumstance, the remote client program sends a record that is structured differently than the local service's VIEW, X_C_TYPE, or X_COMMON buffer. For example, the structure members of the input record may be sequenced differently than the structure members of the input buffer. |
| The remote client program requires a reply record that differs in both type and structure from the reply buffer the local service provides. | In this case, the remote client program requires a particular record and the local service provides a corresponding BEA Tuxedo FML buffer. |

You can omit the INRECTYPE parameter if the local server program sends a reply buffer that is identical in type and structure to the reply record the remote client expects.

## Guidelines for Mapping Output Buffers to Output Records

The following sections provide detailed information explaining how to use the OUTBUFTYPE and OUTRECTYPE parameters for service calls that originate on remote computers (where remote client programs call local services and receive output from those services).

OUTBUFTYPE

The OUTBUFTYPE parameter specifies the request buffer type that the local TMA TCP Gateway provides to the local server.

OUTRECTYPE

The OUTRECTYPE parameter is used to specify the type, and in some cases the format, of the request record a particular remote client program sends to the TMA TCP Gateway. The TMA TCP Gateway uses this information to convert request records from remote clients into buffers that local server programs can process.

You must specify the OUTRECTYPE parameter when one of the circumstances described in the following table is true.

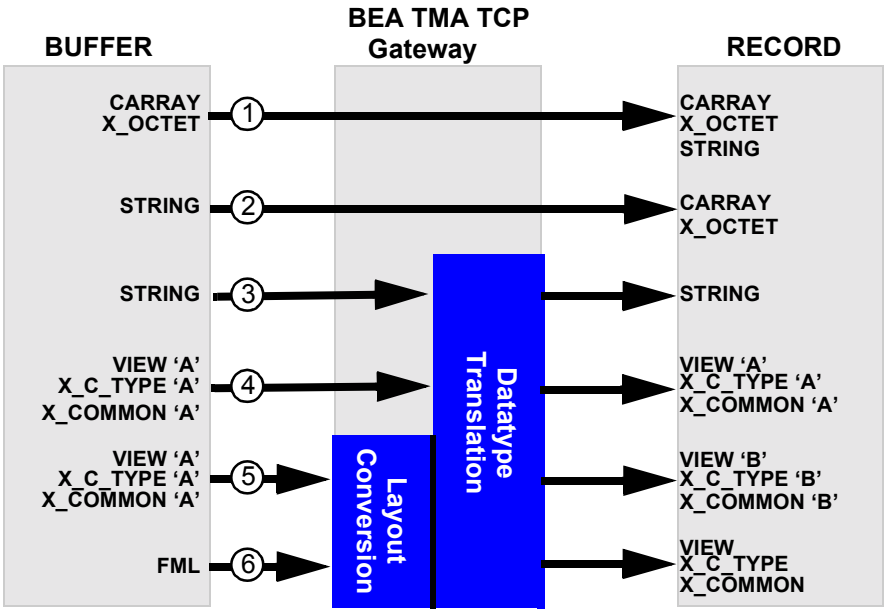
| Circumstance | Explanation |
|---|---|
| The remote client program provides a request record that is structured differently than the local service's request buffer. | In this circumstance, the remote client program provides a record that is structured differently than the local service's VIEW, X_C_TYPE, or X_COMMON buffer. For example, the local server program may expect structure members to be sequenced differently. |
| The remote client program provides an request record that differs from the local service's request buffer in both type and structure. | In this case, the remote client outputs a record and the local client program expects a corresponding BEA Tuxedo FML buffer. |

The OUTRECTYPE parameter may be omitted if the local service's request buffer is identical, in type and structure, to the request record the remote client program provides.

# Mapping Buffers to Records

The following figure shows all the possibilities for mapping buffers to records. The TMA TCP Gateway is responsible for mapping buffers to records, based on information it finds in the TMA TCP Gateway configuration. This mapping occurs for Tuxedo client requests and Tuxedo server responses.

Figure 3-3 Buffer to Record Mappings




# Setting the INBUFTYPE and INRECTYPE Parameters

The following table lists some of the mapping possibilities that are shown in the previous figure and some suggestions for setting the `INBUFTYPE` and `INRECTYPE` parameters.

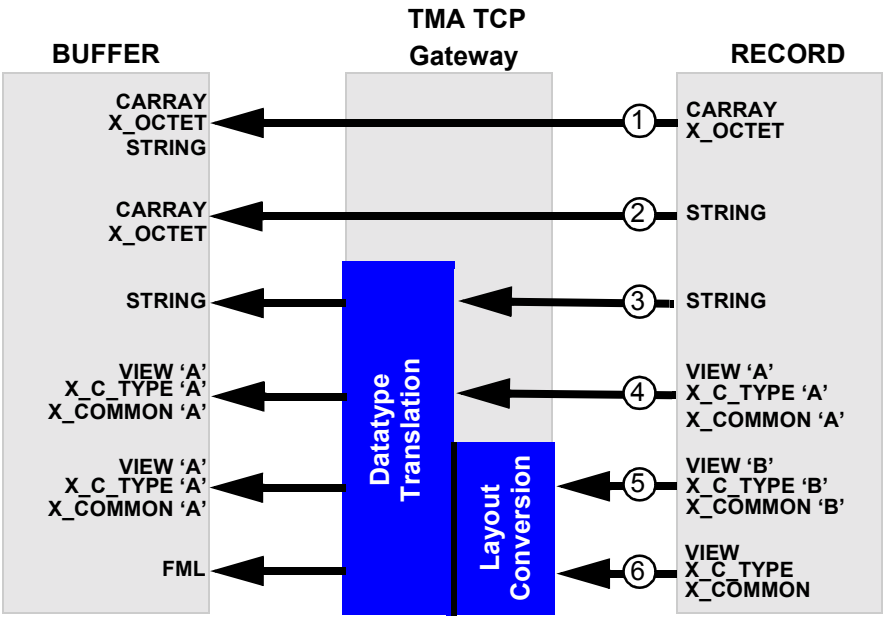
| | |
|---|---|
| 1 | BEA Tuxedo CARRAY input buffers can be copied to CARRAY input records. A CARRAY buffer contains raw data that is not converted or translated. Set the INBUFTYPE parameter to CARRAY, and omit the INRECTYPE parameter.<br><br>CARRAY input buffers can also be copied to STRING input records. This creates a string that goes through no conversion and no translation. The resultant buffer is the length of the original CARRAY buffer. Since all characters are copied, if the CARRAY buffer contains null characters, it affects the buffer when later handled as a STRING. The INBUFTYPE parameter should be set to CARRAY and the INRECTYPE parameter should be set to STRING. |
| 2 | BEA Tuxedo STRING input buffers can be mapped to CARRAY input records. No data conversion or translation is performed. The STRING buffer is copied through the leftmost null character only. Set the INBUFTYPE parameter to STRING and the INRECTYPE parameter to CARRAY. |
| 3 | BEA Tuxedo STRING input buffers can be mapped to STRING input records. The buffer goes through data type translation (such as ASCII to EBCDIC). Set the INBUFTYPE parameter to STRING, and omit the INRECTYPE parameter. |
| 4 | BEA Tuxedo VIEW input buffers can be mapped to identical BEA Tuxedo VIEW input records. In this situation, the data structure that the remote service expects is identical to the data structure the client program uses. There is no need to create a new VIEW definition. Instead, specify the input record type (VIEW) and the name of the existing VIEW definition (the one the client program currently uses) for both the INBUFTYPE parameter, and omit the INRECTYPE parameter. |

| | |
|---|---|
| 5 | BEA Tuxedo VIEW input buffers can be mapped to VIEW input records—in any combination. However, in this situation, the data structure that the remote service expects (designated as VIEW 'B' mapping possibilities in Figure 3-3) differs from the data structure the client program uses (designated as VIEW 'A' in Figure 3-3). Consequently, you must<br>1. Create a VIEW definition for the data structure that the remote service expects.<br>2. Specify the desired record type and the name of this VIEW definition with the INRECTYPE parameter.<br>3. Set the INBUFTYPE parameter to VIEW:*original-viewname*. |
| 6 | Before a BEA Tuxedo FML input buffer can be sent to a remote service that does not support FML, it must be mapped to one of the following input record types: VIEW, X_C_TYPE, or X_COMMON. Also, you must create a VIEW definition for the input data structure that the remote service expects. Set INBUFTYPE to FML and INRECTYPE to VIEW:*viewname*.<br>**Note:** In the DMCONFIG file, if FML or FML32 are specified as INBUFTYPE or OUTBUFTYPE, the type must be followed by a colon (:). (Example: INBUFTYPE="FML32:") |

# Mapping Records to Buffers

The following figure shows all the possibilities for mapping records to buffers. The TMA TCP Gateway is responsible for mapping records to buffers, based on information it finds in the TMA TCP Gateway configuration. This mapping occurs for remote client requests and remote server responses.

**Figure 3-4 Record to Buffer Mappings**




# Setting the OUTRECTYPE and OUTBUFTYPE Parameters

The following table lists some of the mapping possibilities that are shown in the previous figure and some suggestions for setting the `OUTRECTYPE` and `OUTBUFTYPE` parameters (for service calls that originate locally).

| | |
|---|---|
| 1 | BEA Tuxedo CARRAY output records can be copied to CARRAY output buffers. A CARRAY buffer contains raw data that is not converted or translated. Set the OUTBUFTYPE parameter to CARRAY. The OUTRECTYPE parameters need not be set.<br><br>BEA Tuxedo CARRAY output records can also be copied to STRING output buffers. This creates a string that goes through no conversion and no translation. The resultant buffer is the length of the original CARRAY buffer. Since all characters are copied, if the CARRAY buffer contains null characters, it affects the buffer when later handled as a STRING. The OUTRECTYPE should be set to CARRAY and OUTBUFTYPE should be set to STRING. |
| 2 | BEA Tuxedo STRING output records can be mapped to CARRAY output buffers. There is no data conversion or translation performed. The STRING buffer is copied through the leftmost null characters only. Set OUTRECTYPE to STRING and OUTBUFTYPE to CARRAY. |
| 3 | BEA Tuxedo STRING output records can be mapped to STRING output buffers. The buffer goes through datatype translation (such as ASCII to EBCDIC). OUTBUFTYPE and OUTRECTYPE parameters are set to STRING. |
| 4 | BEA Tuxedo VIEW output records can be mapped to identical BEA Tuxedo VIEW output buffers. In this situation, the data structure that the remote service returns is identical to the data structure the local client program expects. There is no need to create a new VIEW definition. Instead, specify the VIEW buffer type and the name of the existing VIEW definition with the OUTBUFTYPE parameter. The OUTRECTYPE parameter can be set to VIEW:*viewname*, but it is not mandatory. |

| | |
|---|---|
| 5 | BEA Tuxedo VIEW output records can be mapped to VIEW output buffers—in any combination. However, in this situation, the data structure that the remote service returns (designated as VIEW 'B' in Figure 3-4) differs from the data structure the client program expects (designated as VIEW 'A' in Figure 3-4). To facilitate the conversion process, perform the following tasks.<br>1. Create a VIEW definition for the data structure that the remote service returns.<br>2. If the name given to the VIEW definition is different than the name that the remote service returns (that is, ATMI buffer subtype), specify the output record type and the name of VIEW 'B' with the OUTRECTYPE parameter. (By doing this, you override the value the TMA TCP Gateway requester automatically detects.)<br>3. Specify the output buffer type and the name of an existing view (VIEW 'A' in the figure) specified in the OUTBUFTYPE parameter. |
| 6 | BEA Tuxedo VIEW output records can be mapped to FML output buffers. To facilitate the conversion process, you must perform the following tasks.<br>1. Create a VIEW definition that describes the data structure that the remote service returns.<br>2. If the name given to the VIEW definition is different than the name that the remote service returns (that is, the ATMI buffer subtype), specify the output record type and the name of your VIEW definition with the OUTRECTYPE parameter. (By doing this, you override the value the TMA TCP Gateway requester automatically detects.)<br>3. Set the OUTBUFTYPE parameter to FML.<br>**Note:** In the DMCONFIG file, if FML or FML32 are specified as INBUFTYPE or OUTBUFTYPE, the type must be followed by a colon (:). (Example: INBUFTYPE="FML32:") |

# Creating VIEW Definitions to Facilitate Buffer Conversion

VIEW definitions are used to describe input and output records that are sent to and received from remote systems. They describe data elements and indicate how data elements are typed and sequenced. Based on these descriptions, TMA TCP Gateway translates field data types as required to maintain transparency between dissimilar systems.

You should create VIEW definitions before you configure TMA TCP Gateway. For complete information about VIEW definitions and related topics, see the *BEA Tuxedo Programmer's Guide*.

The TMA TCP Gateway buffer and record conversion capabilities are extremely powerful and flexible. The key to maximizing these capabilities is to thoroughly understand the BEA Tuxedo VIEW definition mechanism.

VIEW definitions make it possible to specify composite data structures that can be used:

- On different kinds of machines
- With different programming languages

## Preparing VIEW Definitions

After determining the input and output record layouts for the remote application programs you are working with, you need to prepare VIEW definitions and specify these definitions in the configuration files.

**Note:** FML fields must be specified for all VIEWs that TMA TCP Gateway converts. In other words, any VIEW that you specify as an INRECTYPE, OUTRECTYPE, INBUFTYPE, or OUTBUFTYPE must be defined with appropriate FML fields (no dashes in the FNAME column of the VIEW definition). For the FML fields to match, you must compile these VIEWs without the -n option specified.

1. Create standard BEA Tuxedo VIEW definitions in files.
2. Run the viewc or viewc32 VIEW compiler.
3. Set the VIEWFILES, VIEWDIR, FIELDTBLS, and FLDTBLDIR environment variables, using a BEA Tuxedo ENVFILE if necessary (so that TMA TCP Gateway servers can locate binary VIEW files and field table files at runtime).
4. After these tasks are complete, you can specify VIEW definitions in the GWICONFIG and DMCONFIG files (by associating names of VIEW definitions with the INRECTYPE, OUTRECTYPE, INBUFTYPE, and OUTBUFTYPE parameters, as required).

   For detailed information about configuring TMA TCP Gateway, see the "Configuring BEA TMA TCP Gateway" section.

# Translating Data

When a local client program sends data to (or receives data from) a service routine on a different kind of computer, TMA TCP Gateway automatically translates data as required. Translation involves changing the representation of intrinsic data types by changing attributes such as word length and byte order.

The TMA TCP Gateway automatically translates input and output data as required, following rules that are described in the following section. Read the information carefully before you create VIEW definitions (to facilitate buffer conversion) and configure TMA TCP Gateway.

Basic rules for how TMA TCP Gateway translates data are described in the following subsection. For detailed information about how TMA TCP Gateway handles string and numeric data, refer to the "NULL Characters in String Length Calculations (C Programs)" section.

# Data Translation Rules

The following information outlines the data translation rules that TMA TCP Gateway follows:

- `CARRAY` fields are passed untranslated as sequences of bytes.
- `STRING` and `CHAR` fields undergo ASCII-to-EBCDIC translation, if needed.
- `SHORT` and `LONG` fields are translated to S9(4) `COMP` and S9(9) `COMP`, respectively.
- `FLOAT` and `DOUBLE` fields translate to `COMP-1` and `COMP-2`, respectively.
- XML fields undergo ASCII-to-EBCDIC translation, if needed.

**Warning:** `dec_t` cannot be used with `VIEW` translations.

**Note:** BEA Tuxedo provides a field type named `dec_t` that supports decimal values within `VIEW`s. The TMA TCP 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 following table summarizes the relationships.

| Remote Data Type | Description | View Field Type/Length |
|---|---|---|
| `PIC X(n)` | Alphanumeric characters | string / n |
| `PIC X` | Single alphanumeric character | char |
| `PIC X(n)` | Raw bytes | carray / n |
| `PIC 9` | Single numeric byte | carray / 1 |
| `PIC S9(4) COMP` | 16-bit integer | short |
| `PIC S9(9) COMP` | 32-bit integer | long |
| `COMP-1` | Single-precision floating point | float |

| Remote Data Type | Description | View Field Type/Length |
|---|---|---|
| COMP-2 | Double-precision floating point | double |
| PIC S9((m+(n+1))/2)V9(n) COMP-3 | Packed decimal | dec_t / m,n |

# NULL Characters in String Length Calculations (C Programs)

When you create VIEW definitions for input and output buffers that are used by C language applications, you must specify extra characters for terminating NULL characters that are used in string fields.

For example, when a local application program expects a 10-byte string in an output buffer, you would specify 11 for that field—10 for the string plus 1 for the terminating NULL character.

# NULL Characters in String Length Calculations (COBOL Programs)

When you create VIEW definitions for input and output buffers that are used by COBOL language applications, do not specify extra characters for terminating NULL characters that are used in string fields.

For example, when a remote COBOL application program expects 10 characters in an input record, you would specify 10 for that field, not 10 plus 1 (for the terminating NULL character).

**Note:** Although TMA TCP Gateway does not require strings to be NULL-terminated, it respects NULL termination. Therefore, when TMA TCP Gateway 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 TMA TCP Gateway product provides standard character translation from ASCII-to-EBCDIC and EBCDIC-to-ASCII. TMA TCP Gateway automatically performs this translation on the STRING data type.

## Converting Numeric Data

Numeric data can easily be converted into different data types, provided that you have enough range in the intermediate and destination types to handle the maximum value you need to represent.

For example, you can convert numeric values into strings (and the reverse). 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.

# Encoding COBOL Data Types

An additional encoding library, `ConvMVSC`, has been included for Tuxedo clients using COBOL data types. This library, `ConvMVSC`, is similar to the default library, `ConvMVS`, but differs in the following ways:

- For the `STRING` data type, both `ConvMVSC` and `ConvMVS` perform ASCII-to-EBCDIC and EBCDIC-to-ASCII translation.
  - For strings sent to a remote gateway, both libraries perform ASCII-to-EBCDIC conversion, and forward the string to the mainframe.
  - For strings received from a remote gateway, the `ConvMVS` library performs EBCDIC-to-ASCII translation, truncates any trailing space characters and adds a NULL terminator. The `ConvMVSC` library performs the translation, but does not truncate spaces or add the terminator.
- For the `VIEW` data type, the `ConvMVS` and `ConvMVSC` libraries treat all field types except `STRING` the same.
  - For `STRING` fields within a view sent to a remote gateway, `ConvMVS` performs ASCII-to-EBCDIC conversion and appends space to ‘pad’ the string with space characters to the size of the field. `ConvMVSC` performs the character conversion, but does not perform any ‘padding’.
  - For `STRING` fields within a view received from a remote gateway, `ConvMVS` performs EBCDIC-to-ASCII conversion, truncates any trailing space characters, and adds a NULL terminator. The `ConvMVSC` library performs the character conversion, but does not truncate spaces or add the terminator.

## Using the COBOL Data Encoding Library

There are two methods for enabling the COBOL data encoding library:

- COBOL data encoding for all services
- COBOL data encoding for messages to and from a specific host

## Encoding for All Services

To enable COBOL data encoding for every service in the gateway, set the parameter `DFLTTYPE="MVSC"` in the `GLOBAL` section of the `GWICONFIG` file.

**Listing 3-1 Encoding for All Services**

```
*GLOBAL

DFLTTYPE=”MVSC”
```

## Encoding Messages To and From a Specific Host

To enable COBOL data encoding for messages to and from a specific host, set the parameter `TYPE="MVSC"` for that hosts `FOREIGN` entry in the `GWICONFIG` file.

**Listing 3-2 Encoding for Messages To and From a Specific Host**

```
*FOREIGN

HOST_NAME
TYPE=”MVSC”
```

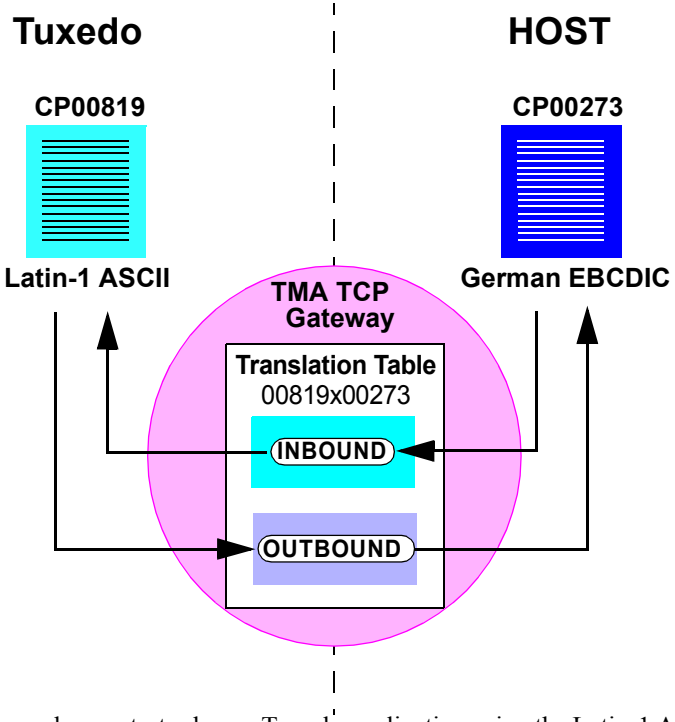
# Using Code Page Translation Tables

The TMA TCP software includes translation tables which enable conversions of single byte character sets (`SBCS`) between ASCII and EBCDIC. These tables are based on IBM-defined code sets and include the default Tuxedo code page, which is the default code page translation table used in previous releases of TMA TCP.

Each translation table consists of two mapping tables, one for outbound conversions (Tuxedo to mainframe) and one for inbound conversions (mainframe to Tuxedo). You do not have to specify

the direction of a translation; however, you must determine the national language in which the host application is written. The following figure illustrates code page translation.

**Figure 3-5 TMA TCP Code Page Translation**




The figure demonstrates how a Tuxedo 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 TMA TCP 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 DMCONFIG file definition for each remote domain. Use the CODEPAGE parameter in the DM_REMOTE_DOMAINS section of the DMCONFIG file. Specify the translation table to use.

To specify a default code page translation for remote hosts to use, specify the translation table filename in the CODEPAGE parameter for the local gateway entry in the DM_LOCAL_DOMAINS section of the DMCONFIG file.

The following table lists the translation tables provided with the TMA TCP software.

**Table 3-1 TMA TCP Code Page Translation Tables**

| Country | File Name | ASCII Code Set | EBCDIC Code Set |
|---|---|---|---|
| Tuxedo default | TUXEDO | TUXEDO-ASCII | TUXEDO-EBCDIC |
| United States | 00819x00037 | CP-00819 | 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 |

**Note:** The Tuxedo default ASCII and EBCDIC code pages differ slightly from CP-00819 and CP-00037.

## How the Translation Tables Work

At start up, the TMA TCP 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 Tuxedo tables, which are hard-coded. You must restart the gateway to change any translation table definitions. The TMA TCP translation tables are located in `$TUXDIR/udataobj/codepage`. For table contents, refer to the "Code Page Translation Tables" section.

If no `CODEPAGE` specification is made for a remote domain, the TMA TCP Gateway software uses the Tuxedo default translation tables.

**Note:** Copies of the default Tuxedo translation tables are included with your product software in `$TUXDIR/udataobj/codepage`. These copies are provided for you to apply modifications if necessary for your applications. These copies are not the actual default tables used by the gateway. You cannot modify the default Tuxedo tables because they are hard-coded.

## Troubleshooting Translation Table Errors

The following information assists you in resolving errors associated with translation tables that cause the gateway to fail. The gateway issues the following message when encountering an error associated with the specified translation table.

`1072:ERROR Cannot read CODEPAGE <`*`filename`*`> for <LOCAL | REMOTE> DOMAIN <`*`domainname`*`>`

For a description of error messages, refer to the "Error and Information Messages" section. The following causes may be responsible for the gateway issuing the previous error.

- The gateway cannot find the translation table file.

  Verify that you specified the correct codepage name in the `CODEPAGE` parameter in the `DM_REMOTE_DOMAIN` section of the `DMCONFIG` file and that the file resides in `$TUXDIR/udataobj/codepage`.

- The software cannot read the translation table file.

  Verify that the file name specified for the `CODEPAGE` parameter is valid. Also, verify that the specified file is not corrupt.

- The software cannot read the translation table file because of access permissions.

  Verify that the specified translation table file in the `$TUXDIR/udataobj/codepage` directory has read permissions.

## Sample DMCONFIG Definition for ASCII to EBCDIC Translations

The following listing shows entries defining one local domain (`CIXA`) and two remote domains (`CISA` and `IMSA`). In the following example, it is assumed that the local domain uses ASCII code page CP-00819 and the two remote domains use the German and French EBCDIC code pages CP-00273 and CP-00297, respectively.

**Listing 3-3 Code Page Definition Example**

```
# DMCONFIG

*DM_LOCAL_DOMAINS
CIXA
     TYPE=IDOMAIN


*DM_REMOTE_DOMAINS
CISA
     TYPE=IDOMAIN
     CODEPAGE=00819X00273
IMSA
     TYPE=IDOMAIN
     CODEPAGE="00819X00297"
```

CHAPTER 4

# Setting Up Security for BEA TMA TCP Gateway

The Tuxedo Mainframe Adapter for TCP Gateway (hereafter referenced as TMA TCP Gateway) component supports security features that allows a requester from Tuxedo to pass a user ID requirement through the OTMA or CICS server interfaces for verification through system security, such as RACF.

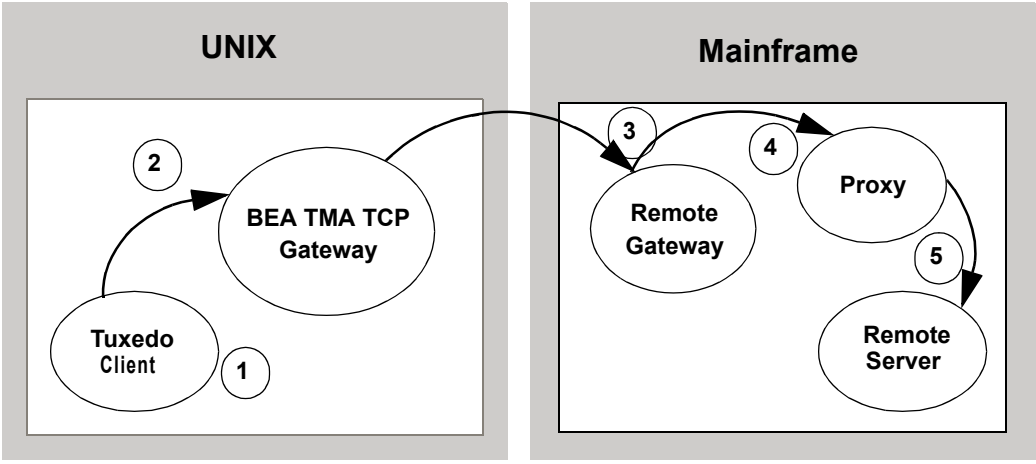
This document explains the following security topics:

- Security Checking from Tuxedo to Mainframe
- Security Checking from Mainframe to Tuxedo
- Setting Up Security
- Sample Security Files
- Data Area Security

## Security Checking from Tuxedo to Mainframe

The following figure illustrates the process flow for security verifications from TMA TCP Gateway to a mainframe.

**Figure 4-1 Security Checking for Tuxedo to Mainframe Transactions**




1. When the client program performs a `tpinit()`, the user's Tuxedo identity is validated against the `tpusr` file.
2. When the client program issues a `tpcall()` or `tpacall()`, Tuxedo verifies (against the `tpacl` file) that the user is authorized to invoke the gateway service.
3. When the gateway establishes the initial connection, connection security information (specified as `RMTNAME` and `PASSWORD` in the `GWICONFIG` file) is passed from the TMA TCP Gateway to the remote gateway. If the `RMTNAME` and `PASSWORD` values match the values configured on the remote gateway, the connection is established.

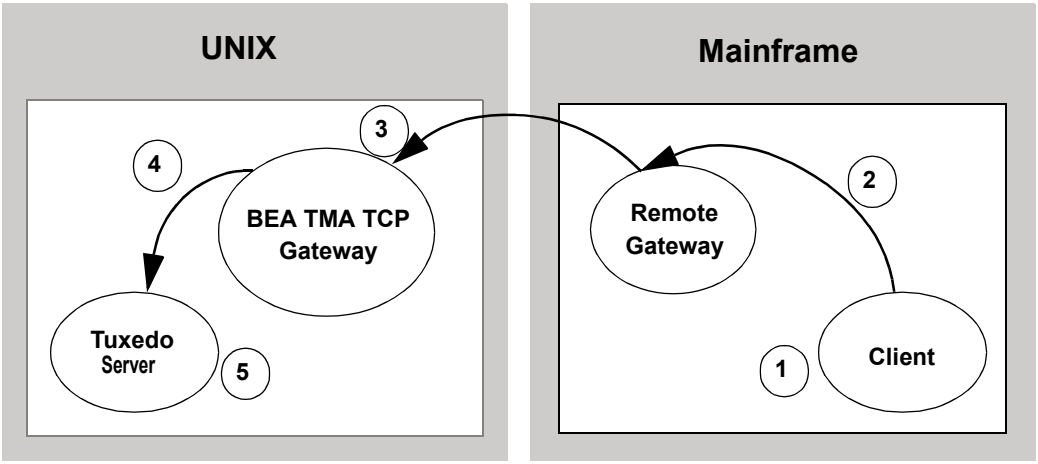
   With each request, the TMA TCP Gateway passes the user's Tuxedo identity to the remote gateway.

   **Note:** To pass authority checking, the user's Tuxedo identity must match the mainframe user ID exactly.
4. The remote mainframe gateway initiates a proxy to act on behalf of the specified user ID.
5. The proxy calls the specified service using system security to check authorization.

# Security Checking from Mainframe to Tuxedo

The following figure illustrates the process flow for security verifications from a mainframe to TMA TCP Gateway.

**Figure 4-2 Security Checking for Mainframe to Tuxedo Transactions**




1. The user ID, established at mainframe log in, is checked by system security to verify that the user has permission to start a client transaction.
2. The user ID is checked by system security to verify that the user has permission to send a request to the gateway.
3. With each request, the gateway passes the user ID to the Tuxedo gateway.

   **Note:** To pass authority checking, the user's Tuxedo identity must match the mainframe user ID exactly.
4. The TMA TCP Gateway maps the mainframe user ID to a Tuxedo user ID and issues the service request on behalf of that user.
5. The Tuxedo server performs access checks (based on the `tpacl` file) to verify that the user has access to the requested service.

# Setting Up Security

The TMA TCP Gateway product supports two methods for providing security:

- Tuxedo security plug-in
- Built-in Tuxedo security

# Tuxedo Security Plug-in

The Tuxedo security plug-in enables the customization of the security functions, including the use of alternate implementations. The Tuxedo security plug-in is set up during Tuxedo plug-in configuration. Refer to the Tuxedo documentation for specific information about this feature.

# Built-in Tuxedo Security

Built-in Tuxedo security is used when custom security plug-ins are not implemented.

To enable the built-in Tuxedo security feature, complete the following tasks.

1. Code `SECURITY` in the BEA Tuxedo `UBBCONFIG` file. Refer to the *BEA Tuxedo Administration Guide* for more information.
2. Set up user, group, and `ACL` files. Refer to the *BEA Tuxedo Administrator's Guide* for more information.

   **Note:** The user information in these files must match in the BEA Tuxedo and the mainframe environments or a security violation occurs.
3. Code the security parameter in your TMA TCP Gateway configuration file (`GWICONFIG`). For `GWICONFIG` syntax and parameter definitions, refer to the "Configuring BEA TMA TCP Gateway" section.

## Sample Security Files

Part of the process for setting up security for TMA TCP requires you to have user, group, and `ACL` files. The following sections include these sample files.

### User Files

The following sample is a user file that includes user names, encrypted passwords, a user ID number, group number, and a client name.

**Listing 4-1 Sample User (tpusr) File**

```
#illen:w2ZMOKeJmiU0M:1:0:TPCLTNM,someguy::
#illen:0YzvQeqzcNz56:1:0:TPCLTNM,*::
#eke:x3vG37eOqh0XE:2:0:TPCLTNM,*::
#illen:0YzvQeqzcNz56:1:1:TPCLTNM,*::
#illen:0YzvQeqzcNz56:1:2:TPCLTNM,*::
```

```
john:x3vG37eOqh0XE:2:1:TPCLTNM,*::
jim:0YzvQeqzcNz56:1:1:TPCLTNM,*::
richard:IxqosKHu5Q3BA:3:1:TPCLTNM,*::
JDOE:zBMWVUBNNBVgo:4:0:TPCLTNM,*::
smith:ULfRJzAeyGAD2:5:0:TPCLTNM,*::
```

Lines that begin with the pound sign (#) are users that have been changed or deleted by `tpusrmod` or `tpusrdel`.

## Group File

The following sample is a group file that specifies the names and indexes of groups.

**Note:** The `tpgrp` file is only necessary when specifying `ACL` or `MANDATORY_ACL` modes for security. If you specify `USER_AUTH` for security, you can assign users to groups, but they do not correlate to the groups used for security by the remote system.

**Listing 4-2 Sample Group (`tpgrp`) File**

```
good::1:
bad::2:
```

## ACL File

The `tpacl` file correlates a group and the services to which that group has access. In the `tpacl` file, the first field specifies what is protected, the second field specifies the type of object being protected (specified in the first field), and the third field specifies the group that has access to the object.

In the following example, only users in group 1 (john, jim, richard) can access `TOLOWER`, and only users in group 2 can access `TOUPPER`.

**Note:** The `tpacl` file is only necessary when specifying `ACL` or `MANDATORY_ACL` modes for security.

**Listing 4-3 Sample ACL (tpacl) File**

```
TOLOWER:SERVICE:1:
TOUPPER:SERVICE:2:
```

# Data Area Security

The TMA TCP Gateway provides data area security which is a specialized security protocol for the following cases:

- User information is propagated across multiple Tuxedo domain boundaries
- A remote or local service requires a user's LTERM information

In these cases, a client's user ID, group name, and LTERM can be specified in the data area of a request. For Tuxedo clients, user information specified in the data area is verified by the remote gateway in the usual manner. For remote clients, remote user information is placed in the data area fields by the local gateway to be used by Tuxedo services. In this case, the remote client does not have to populate these fields, but must allocate space for them in the data area.

## Enabling Data Area Security

Complete the following tasks to enable data area security.

1. Add fields to the user's data area on the local and remote hosts. These fields are passed to and from the mainframe host. For the field formats, refer to Listing 4-4.
2. Set WRAP=TPSD in the FOREIGN section corresponding to the remote host in the GWICONFIG file. For syntax and parameter definitions for the FOREIGN section of the GWICONFIG file, refer to the "Defining the FOREIGN Section of the GWICONFIG File" section.
3. Populate the data area with the user information before sending a request to a remote service.
4. The remote user's information is populated into the data area when a request is received for a local service.

   **Note:** If using a VIEW data format, allocate the extra fields before the application data as defined in Listing 4-4. If using the STRING data format, allocate 24 additional bytes at the beginning of the string to be used for the security fields.

# Format

The user data area fields in C use the following format.

**Listing 4-4 Syntax for C User Data Area Fields**

```
struct da_security {
       char uname[8];  /*user name*/
       char group[8];  /*user group*/
       char lterm[8];  /*terminal id*/
       /*user data is appended here*/
}
```

CHAPTER 5

# Configuring BEA TMA TCP Gateway

The following configuration files must be set up prior to running BEA Tuxedo Mainframe Adapter for TCP Gateway (hereafter referenced as TMA TCP Gateway):

- UBBCONFIG
- GWICONFIG
- DMCONFIG

This document explains the following tasks for configuring TMA TCP Gateway:

- Updating the BEA Tuxedo UBBCONFIG File
- Specifying Parameters in the GWICONFIG File
- Defining Domain Configurations in the DMCONFIG File

## Updating the BEA Tuxedo UBBCONFIG File

As with any Tuxedo server, you must establish a server group for TMA TCP Gateway by adding entries for TMA TCP Gateway in the UBBCONFIG file for the local region. Specifically, you must add records to:

- The GROUPS section
- The SERVERS section
- The SERVICES section

See the *BEA Tuxedo Administrator's Guide* for more information about the UBBCONFIG file. For information about the UBBCONFIG file specific to the gateway, refer to the "Updating the GROUPS Section to Establish a Server Group" and "Updating the SERVERS Section" sections.

**Note:** Lines beginning with an asterisk (*) indicate the beginning of a specification section. Each such line contains the name of the section immediately following the *. The asterisk is required when specifying a section name.

# Updating the GROUPS Section to Establish a Server Group

To establish a server group for TMA TCP Gateway, you must add the following items to the GROUPS section of the BEA Tuxedo UBBCONFIG file:

- A name for the TMA TCP Gateway server group using the *groupname* variable
- The logical machine name for the system on which TMA TCP Gateway is installed using the LMID parameter

  **Note:** For Windows NT, the LMID must be in uppercase
- A group number for the TMA TCP Gateway server group using the GROUPNO parameter

## Syntax

The syntax of the configuration file entry is as follows.

**Listing 5-1 Syntax for GROUPS Section of UBBCONFIG File**

```
groupname LMID=logical_machine_identifier
          GROUPNO=group_number
```

The variable definitions follow.

| Variable | Description |
|---|---|
| *groupname* | Specifies the name you select for the TMA TCP Gateway server group. |

| Variable | Description |
|---|---|
| *logical_machine_identifier* | Specifies the logical machine identifier for the system on which TMA TCP Gateway is installed. |
| *group_number* | Specifies the number you assign to the TMA TCP Gateway server group. |

## Example

Here is an example of a UBBCONFIG entry that establishes a server group.

**Listing 5-2 Establishing a Server Group**

```
NODE2GATE    LMID=NODE2
             GROUPNO=1
```

# Updating the SERVERS Section

This section explains how to specify TMA TCP Gateway servers in the BEA Tuxedo configuration.

The TMA TCP Gateway product provides the gateway you need to list in the SERVERS section of the BEA Tuxedo UBBCONFIG configuration file.

For each TMA TCP Gateway server group, you can specify one gateway. There must also be entries for domain administration (DMADM) and gateway administration (GWADM) servers.

## Syntax

The syntax of each configuration file entry is as follows.

**Listing 5-3 Syntax for SERVER Section of UBBCONFIG File**

```
DMADM       SVRGRP=groupname SRVID=integer
GWADM       SVRGRP=groupname SRVID=integer
```

```
GWIDOMAIN  SVRGRP=groupname SRVID=integer
           CLOPT="-A -- -r"
```

The following table describes the parts of the syntax.

| Part | Description |
|---|---|
| GWIDOMAIN | Specifies the name of the TMA TCP Gateway. |
| DMADM | Specifies the name of the BEA Tuxedo supplied domain administration server. |
| GWADM | Specifies the name of the BEA Tuxedo supplied gateway administration server. |
| *groupname* | Specifies the name you select for the TMA TCP Gateway server group. This is the group with which the server in question is associated. |
| *integer* | Specifies the number you assign to the TMA TCP Gateway server. |
| CLOPT="-A -- -r" | -A specifies the command line option string that is used to initialize TMA TCP Gateway when it is invoked. This is the default for BEA Tuxedo server processes.<br>-r after the double dashes specifies that the server should record, in its standard error file, a log of services performed. This log may be analyzed by the txrpt command. When using this option, ensure that the ULOGDEG variable is not set to "y". |

For more information about BEA Tuxedo servers and related configuration parameters, see the BEA Tuxedo *Administrator's Guide*.

## Using the Request Logging Option

Output from the -r command line option should be capable of being processed by txrpt command. The txrpt command analyzes the standard error output of a BEA Tuxedo server and provides a summary of service processing time within the server. The report shows the number of times each service was dispatched and the average amount of time it took for each service to

process a request during the specified period. `txrpt` takes its input from the standard input or from a standard error file redirected as input.

**Notes:** The process of gathering statistics creates overhead. Use this option selectively.

The -r option applies to servers but does not apply to GWIDOMAIN because it is a Gateway.

### Other Options for Configuring Servers

As with other Tuxedo servers, you can use some of BEA Tuxedo system server boot options with TMA TCP Gateway servers. Boot options must be specified with `CLOPT` parameter (before the `CLOPT` double-dash separator).

**Note:** Because TMA TCP Gateway dynamically advertises services that are listed in its initialization file, you should not advertise services by using the `-s` option.

For more information about these and other boot options, see `servopts(5)` in the BEA Tuxedo *Administrator's Guide*.

## Specifying Parameters in the GWICONFIG File

The `GWICONFIG` file is the mechanism that system administrators use to configure TMA TCP Gateway. The particular file the gateway uses is determined by the environment variable `GWICONFIG`. The configuration file is similar to the Tuxedo Transaction Manager `UBBCONFIG` file, both in structure and in composition. This makes it possible for experienced BEA Tuxedo administrators to configure TMA TCP Gateway without extensive training.

**Note:** `GWICONFIG` is a generic filename. You are free to choose other filenames just as with the Tuxedo `UBBCONFIG` file and `DMCONFIG` file. Be sure that the name of the `GWICONFIG` file is specified in the `GWICONFIG` environment variable. Also, the `GWICONFIG` file should be saved to the application directory.

The `GWICONFIG` file is divided into the following required sections:

**Table 5-1 Required GWICONFIG File Sections**

| Section | Description |
|---|---|
| GLOBAL | Describes certain general characteristics of all TMA TCP Gateways. |
| NATIVE | Describes all native systems. Use the gateway name, specified in the GWI_GWNAME environment variable, to distinguish different gateways in the same configuration file. |
| FOREIGN | Describes all foreign systems. |
| LOCAL_SERVICES | Describes local services that are accessible in the BEA Tuxedo domain through the TMA TCP Gateway. Each local service is linked to a native system using the NATIVE parameter. For each local service, TMA TCP Gateway uses the TCP port number and IP address of the corresponding native system to establish a TCP listening endpoint. |
| REMOTE_SERVICES | Describes remote services that are accessible from the BEA Tuxedo domain. Each remote service describes a TCP service on a foreign system. For each remote service, the FOREIGN parameter defines the name of the foreign system. TMA TCP Gateway uses the TCP port number and IP address of the foreign system to connect to the remote TCP service. |

**Warning:** The GWICONFIG file *must* contain the required sections in the order shown in Listing 5-4. Each section requires the asterisk (*) in the name.

**Listing 5-4 Sample GWICONFIG File**

```
*GLOBAL
                NWDEVICE="/dev/tcp"
                CONNECT_TIME=20
                OUTREQ_TIME=20
                LATENCY=-2
                SECURE=N
                MULTIPLEX=2
                DFLTWRAP="TPS"
                DFLTTYPE="MVS"
*NATIVE
LOCAL           IPADDR="//beasun2"
```

```
                TCP_PORT=9002
                IDLE_TIME=20

*FOREIGN
RIGHTY          WRAP="TPS"
                TYPE="MVS"
                IPADDR="//beasun2"
                TCP_PORT=9004
                MULTIPLEX=2
                IDLE_TIME=30
                RMTACCT="zeke"
                PASSWORD="maple"

MIKE            WRAP="TPS"
                TYPE="MVS"
                IPADDR="//dalvs3"
                TCP_PORT=9001
                MULTIPLEX=6

*LOCAL_SERVICES
TUXTOUPPER

ECHO

*REMOTE_SERVICES
TUXTOLOWER
                OUTREQ_TIME=20

BEASVR07
                OUTRECTYPE="VIEW:weird"

TST1V
```

**Note:** Changes to the GWCONFIG file must be made when upgrading from previous releases of TMA TCP. See the BEA Tuxedo Mainframe Adapter for TCP *Release Notes* for specific upgrade information.

# Defining the GLOBAL Section of the GWICONFIG File

The following sections describe parameters that are associated with the GLOBAL section of the GWICONFIG file. These parameters describe global characteristics for the gateway.

The format of the GLOBAL section of the GWICONFIG file is shown in the following listing.

**Listing 5-5 Syntax for GLOBAL Section of GWICONFIG File**

```
[ # A comment (from "#" to end of line)]

*GLOBAL
[ NWDEVICE=TCP_device ]
[ CONNECT_TIME=n]
[ IDLE_TIME=n]
[ OUTREQ_TIME=n]
[ LATENCY=n|1 ]
[ SECURE=Y|N ]
[ MULTIPLEX=n|1 ]
[ DFLTWRAP=wrapper name ]
[ DFLTTYPE=translation type ]
```

The following table describes the parameters that are set in the GLOBAL section.

**Table 5-2 Parameters for the `Global` Section**

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| `NWDEVICE="`*`TCP_device`*`"` | Optional | `"/dev/tcp"` | Specifies the network device to be used for communication with remote gateways. |
| `CONNECT_TIME=`*`n`* `seconds` | Optional | No timeout | Specifies the number of seconds the gateway waits to establish a connection. |
| `IDLE_TIME=`*`n`* `seconds` | Optional | No idle timeout | Specifies the number of seconds a connection can be idle before timing out. |
| `OUTREQ_TIME=`*`n`* `seconds` | Optional | None | Specifies the default timeout value, in seconds, for requests sent to foreign gateways. |
| `LATENCY=`*`n`* `seconds` | Optional | `1` | Specifies the number of seconds to be deducted from the timeout value sent to a remote gateway for a Tuxedo client request. This increases the likelihood that the remote gateway detects a timeout before the local gateway. |

**Table 5-2 Parameters for the `Global` Section**

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| `SECURE=Y \| N` | Optional | `N` | Specifies whether the TMA TCP Gateway supplies user information to local and remote services.<br>If `SECURE=Y`, then the TMA TCP Gateway supplies user information to remote services and applies remote user information to local services using the `appkey`.<br>If `SECURE=N`, then the TMA TCP Gateway does not supply user information to remote services and does not apply remote user information to local services using the `appkey`.<br>If `SECURE=N` and the Tuxedo domain is set with `SECURITY=ACL` in the `UBBCONFIG` file, a request to a local service can fail even if ACLs are in place. |
| `MULTIPLEX=`*n* | Optional | `1` | Specifies the maximum number of outstanding requests per connection that the local gateway can support. |

**Table 5-2 Parameters for the `Global` Section**

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| `DFLTWRAP=`*`wrapper name`* | Optional | `TPS` | Specifies the name of the default wrapping library to use for wrapping and unwrapping messages for machines without a `FOREIGN` section or without a `WRAP` parameter in the `FOREIGN` section. A corresponding wrapper object `WRAP<`*`wrapper name`*`>` must exist. The wrapper name `TPS` is used in most cases. Specify `TPSD` if data area security is used. For more information about data area security, refer to "Setting Up Security for BEA TMA TCP Gateway." |
| `DFLTTYPE=system type` | Optional | `DFLTYPE=MVS` | Specifies the default foreign system type for encoding and decoding message buffers. If you specify `TYPE` in the `FOREIGN` section, that `TYPE` definition overrides this default value. For normal C-to-COBOL encoding, specify `DFLTYPE="MVSC"` to enable COBOL data encoding.<br>For more information about `DFLTYPE` values MVS and MVSC, refer to "Configuring BEA TMA TCP Gateway for Data Mapping." |

# Defining the NATIVE Section of the GWICONFIG File

The following sections describe parameters that are associated with the `NATIVE` section of the `GWICONFIG` file. These parameters are specific to the local system. You can specify multiple native systems in the same configuration file allowing multiple gateway processes to access the same configuration file. This makes a single repository of connectivity services. The link between the gateway process and the native system entry is made through the `GWINAME` environment variable.

The format of the `NATIVE` section of the `GWICONFIG` file is illustrated in the following listing.

**Listing 5-6 Syntax for NATIVE Section of GWICONFIG File**

```
[ # A comment (from "#" to end of line)]
*NATIVE
  <GATEWAY_NAME>
[ IPADDR=ip_address ]
[ TCP_PORT=port number ]
[ IDLE_TIME=n]
[ MULTIPLEX=n]
[ POLL_TIME=n]
[ MAXCONNECT=n]
```

The following table describes for each local system that are specified in the NATIVE section.

**Table 5-3 Parameters for the NATIVE Section**

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| <*GATEWAY_NAME*> | Required | None | This parameter is a 1-78 alphanumeric character string that represents the gateway identification passed in the GWINAME environment variable.<br>The *GATEWAY_NAME* parameter must match an entry in the DM_LOCAL_DOMAINS section of the DMCONFIG file. |
| IPADDR=*ip_address* | Optional | IP address of the local host machine | Specifies the IP address for the local system. The IPADDR can be in the hexadecimal format 0xaaaaaaaa, the dotted decimal format //#.#.#.#, or the DNS format //host[.domainname]. |
| TCP_PORT=*port number* | Optional | None | Specifies the local port number used for listening for services. This parameter is optional if no local services are advertised. If you do not specify TCP_PORT, a listener port is not created. |

**Table 5-3 Parameters for the NATIVE Section**

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| IDLE_TIME=*n* | Optional | No idle time value | Specifies the number of seconds a connection can remain idle before being disconnected. |
| MULTIPLEX=*n* | Optional | 1 | Specifies the number of outstanding requests per connection that the local gateway can support. |
| POLL_TIME=*n* | Optional | 250,000 micro seconds | Specifies the polling timeout (in microseconds) to be used in polling for Tuxedo messages. The range of values for this parameter is 100,000-10,000,000. |
| MAXCONNECT=*n* | Optional | No maximum | Specifies the number of connections into the gateway from remote hosts. If the remote system attempts more connections than this parameter specifies, the remote systems are disconnected. |

# Defining the FOREIGN Section of the GWICONFIG File

The FOREIGN section of the GWICONFIG file contains parameters that collectively describe foreign systems.

The format of the FOREIGN section of the GWICONFIG file is illustrated in the following listing.

**Listing 5-7 Syntax for FOREIGN Section of GWICONFIG File**

```
[ # A comment (from "#" to end of line)]
*FOREIGN
  <SYSTEM_NAME>
  IPADDR=ip_address
  [TYPE=system_type]
  [WRAP=wrapper name]
  [TCP_PORT=port number]
  [MULTIPLEX=n sessions]
```

```
[IDLE_TIME=n seconds]
[RMTACCT="userid"]
[PASSWORD="password"]
[CICS=Y | N]
[CICSHAND=<name>]
[MAXCONNECT=n]
[CONNSYNC=Y|N]
[CONNECT_TIME=n]
[CICSDATA="string"]
```

The following table describes the parameters that are set for the each foreign system that you specify in the FOREIGN section.

The following table describes the parameters that are specified in the FOREIGN section.

**Table 5-4 Parameters for the FOREIGN Section**

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| <*SYSTEM_NAME*> | Required | None | This parameter is a 1-78 alpha-numeric character string that represents the foreign system name.<br>The *SYSTEM_NAME* parameter must match an entry in the DM_REMOTE_DOMAINS section of the DMCONFIG file. |
| IPADDR=*ip_address* | Required | None | Specifies the IP address for the remote system. The IPADDR can be in the hexadecimal format 0xaaaaaaaa, the dotted decimal format //#.#.#.#, or the DNS format //host[.domainname]. |

**Table 5-4 Parameters for the FOREIGN Section**

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| TYPE=*system type* | Optional | "MVS" | Specifies the foreign system type for encoding and decoding the Tuxedo buffers (application data). TYPE values of MVS and MVSC support C-to-COBOL or COBOL data encoding. If you do not specify TYPE, the value in DFLTYPE in the GLOBAL section is used.<br>For more information about MVS and MVSC TYPE values, refer to "Configuring BEA TMA TCP Gateway for Data Mapping." |
| WRAP=*wrapper name* | Optional | "TPS" | Specifies the name of the wrapping entry to use for wrapping and unwrapping messages for this host. A corresponding wrapper object wrap<wrapper name> must exist. The wrapper name TPS is used in most cases.<br>Specify TPSD if data area security is used. For more information about data area security, refer to "Setting Up Security for BEA TMA TCP Gateway."<br>If WRAP is not specified in the FOREIGN section, the value for DFLTWRAP in the GLOBAL section is used. |
| TCP_PORT=*port number* | Optional | No port number | Specifies the port number of the foreign gateway. This parameter is optional if remote services are not defined for this foreign system. |
| MULTIPLEX=*n* sessions per connection | Optional | 1 session per connection | Specifies the maximum number of sessions per connection that the local gateway can support. |
| IDLE_TIME=*n* seconds | Optional | No idle timeout | Specifies the number of seconds a connection can remain idle before being disconnected. |

**Table 5-4 Parameters for the FOREIGN Section**

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| CICS=Y \| N | Optional | N | Specifies whether to send control information to the IBM TCP/IP listener for use with the TMA TCP for CICS gateway.<br>If CICS=Y and you are not using IBM TCP/IP or your remote gateway is not TMA TCP for CICS, the transaction does not process correctly. |
| CICSHAND=<*name*> | Optional | BEAH | Specifies the name of the handler transaction to be passed to the IBM TCP/IP listener for use with TMA TCP for CICS. |
| RMTACCT=*"userid"* | Optional | " " | Specifies the user ID for gateway-level security on the foreign system. |
| PASSWORD=*"password"* | Optional | " " | Specifies the password associated with the user ID for gateway-level security on the foreign system. |
| MAXCONNECT=*n* | Optional | No limit | Specifies the maximum number of connections to the specified host. |
| CONNSYNC=Y\|N | Optional | N | Specifies whether to force the gateway to establish connections to the specified host in a synchronous manner. |
| CONNECT_TIME=*n* | Optional | Value from GLOBAL section | Specifies the number of seconds the gateway waits to establish a connection. If you do not specify this parameter, the value of CONNECT_TIME in the GLOBAL section is used. |
| CICSDATA=*"string"* | Optional | " " | Specifies a string to be passed to the IBM TCP/IP listener for use with the TMA TCP for CICS gateway. |

# Defining the LOCAL_SERVICES Section of the GWICONFIG File

The LOCAL_SERVICES section of the GWICONFIG file contains parameters for each local service specified in the DMCONFIG file. Each service entry name matches the remote name of the service in the DM_LOCAL_SERVICES section.

The format of the LOCAL_SERVICES section of the GWICONFIG file is illustrated in the following listing.

**Listing 5-8 Syntax for LOCAL_SERVICES Section of GWICONFIG File**

```
[ # A comment (from "#" to end of line)]
*LOCAL_SERVICES
  <SERVICE_NAME>
[INRECTYPE="foreign_incoming_buffer_type"]
  [OUTRECTYPE="foreign_outgoing_buffer_type"]
  [SECURE=Y | N]
  [CONV=Y | N]
```

The following section describes the parameters set for each service you specify in the LOCAL_SERVICES section.

**Table 5-5 Parameters for LOCAL_SERVICES Section**

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| <*SERVICE_NAME*> | Required | None | This parameter is a 1-78 alphanumeric character string that represents the local service name that matches the service name value in the DM_LOCAL_SERVICES section of the DMCONFIG file. |
| INRECTYPE="*foreign_incoming_buffer_type*" | Optional | None | Specifies the foreign buffer type for replies to remote clients. If you do not specify INRECTYPE, the default is no type. In this case, the type of buffer is unchanged. |

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| OUTRECTYPE=*"foreign_outgoing_buffer_type"* | Optional | Match the INRECTYPE value | Specifies the foreign buffer type for requests from remote clients. If you do not specify OUTRECTYPE, the default is to match the INRECTYPE value. |
| SECURE=Y \| N | Optional | N | Specifies whether the TMA TCP Gateway applies remote user information to local services.<br>If SECURE=Y, then the TMA TCP Gateway applies remote user information to local services using the appkey.<br>If SECURE=N, then the TMA TCP Gateway does not apply remote user information to local services using the appkey.<br>If SECURE=N and the Tuxedo domain is set with SECURITY=ACL in the UBBCONFIG file, a request to a local Tuxedo service can fail even if ACLs are in place. |
| CONV=Y \| N | Optional | N | Specifies whether service is conversational. Conversational mode is not currently supported, so Y returns an error message and the gateway does not start. |

# Defining the REMOTE_SERVICES Section of the GWICONFIG File

The REMOTE_SERVICES section of the GWICONFIG file contains parameters for each remote service specified in the DMCONFIG file. Each service entry name matches the remote name of the service in the DM_REMOTE_SERVICES section.

The format of the REMOTE_SERVICES section of the GWICONFIG file is illustrated in the following listing.

**Listing 5-9 Syntax for REMOTE_SERVICES Section of GWICONFIG File**

```
[ # A comment (from "#" to end of line)]
*REMOTE_SERVICES
  <SERVICE_NAME>
[INRECTYPE="foreign_outgoing_buffer_type"]
  [OUTRECTYPE="foreign_incoming_buffer_type"]
  [OUTREQ_TIME=n]
  [SECURE=Y | N]
  [CONV=Y | N]
```

The following table describes the parameters set for each service you specify in the REMOTE_SERVICES section.

**Table 5-6 Parameters for REMOTE_SERVICES section**

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| <*SERVICE_NAME*> | Required | | This parameter is a 1-78 alphanumeric character string that represents the local service name that matches the RNAME value in the DM_REMOTE_SERVICES section of the DMCONFIG file. If RNAME is not specified, this name matches the Tuxedo service name. |
| INRECTYPE="*foreign_outgoing_buffer_type*" | Optional | None | Specifies the foreign buffer type for requests to remote servers. If you do not specify INRECTYPE, the default is no type. In this case, the type of buffer is changed. |
| OUTRECTYPE="*foreign_incoming_buffer_type*" | Optional | Match the OUTBUFTYPE value | Specifies the foreign buffer type for replies from remote servers. If you do not specify OUTRECTYPE, the default is no type. In this case, the type of buffer is unchanged. |

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| OUTREQ_TIME=*n* | Optional | Value from GLOBAL section | Specifies the timeout value, in seconds, for requests sent to this service. If you do not specify this parameter, the value for OUTREQ_TIME in the GLOBAL section is used. If OUTREQ_TIME is not specified in this section or the GLOBAL section, an error message occurs. |
| SECURE=Y \| N | Optional | N | Specifies whether the TMA TCP Gateway supplies local user information to remote services.<br>If SECURE=Y, then the TMA TCP Gateway supplies user information to remote services.<br>If SECURE=N, then the TMA TCP Gateway does not supply user information to remote services. |
| CONV=Y \| N | Optional | N | Specifies whether the service is conversational. Conversational mode is not currently supported, so Y returns an error message and the gateway does not start. |

# Defining Domain Configurations in the DMCONFIG File

The domain configuration file (DMCONFIG) is made up of 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 *. The asterisk is required when specifying a section name. Allowable section names are: DM_LOCAL_DOMAINS, DM_REMOTE_DOMAINS, DM_LOCAL_SERVICES, DM_REMOTE_SERVICES, DM_ROUTING, and DM_ACCESS_CONTROL.

**Note:** The DM_LOCAL_DOMAINS section must precede the DM_REMOTE_DOMAINS.

The following paragraphs describe the significant parameters within specific sections of the DMCONFIG file that define new gateway configurations.

# 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.

The format of the DM_LOCAL_DOMAINS section of the DMCONFIG file is illustrated in the following listing.

**Listing 5-10 Syntax for DM_LOCAL_DOMAINS Section of DMCONFIG File**

```
LDOM   required parameters  [optional parameters]
```

LDOM is an *identifier* value used to name each local domain and must be unique within a particular configuration. In the description of the DM_LOCAL_SERVICES section, *LDOM* is the identifier that connects local services with a particular gateway group.

The following table describes the parameters that are set in the DM_LOCAL_SERVICES section.

**Table 5-7 Parameters for theDM_LOCAL_SERVICES Section**

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| GWGRP = *identifier* | Required | None | Specifies the name of the gateway server group (the name provided in the TUXCONFIG file) representing this local domain. A one-to-one relationship must exist between a DOMAINID and the name of the gateway server group; each GWGRP must have its own unique DOMAINID. |
| TYPE = *identifier* | Required | None | Is used for grouping local domain into classes. TYPE can be set to TDOMAIN or any other domain gateway type. The TDOMAIN value indicates that this local domain can only communicate with another Tuxedo System/Domain. For use with TMA TCP Gateway, specify TYPE=IDOMAIN. Domain types must be defined in the $TUXDIR/udataobj/DMTYPE file. |
| DOMAINID = *string* | Required | None | 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 31 characters or fewer. |

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| AUDITLOG = *string* | Optional | None | Specifies the name of the audit log file for this local domain. The audit log feature is activated from the dmadmin 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* | Optional | The value of BLOCKTIME parameter specified in the TUXCONFIG file | 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.<br><br>A timeout always implies a failure of the affected request. Notice that the timeout specified for transactions in the TUXCONFIG is always used when the request is issued within a transaction. |

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| CODEPAGE = *table-identifier* | Optional | "DMTLOG" | Specifies the mapping to use with remote hosts that are not specified in the DMCONFIG file. CODEPAGE designates 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. The name of the file, located in the $TUXDIR/udatajobj/codepage directory, is a composite of the code page numbers used for the translation. This parameter specifies the mapping to use with remote hosts that are not specified in the DMCONFIG file.<br><br>For example, specifying 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. For complete character listings, refer to "Code Page Translation Tables." |
| DMTLOGDEV = *string* | Optional | None | 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, 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. |

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| DMTLOGNAME = *identifier* | Optional | "DMTLOG" | 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. The name must be 30 characters or less. |
| DMTLOGSIZE = *numeric* | Optional | 100 pages | 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. |
| MAXDATALEN = *numeric* | Optional | None | 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* | Optional | None | Specifies the maximum number of connections (or dialogs if the domain is of type *OSITP*) allowed per gateway. There is no limit if this parameter is not specified. |
| MAXRDTRAN = *numeric* | Optional | 16 | 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. |
| MAXTRAN = *numeric* | Optional | The value of MAXGTT | 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* | Optional | None | 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 are broken up into packets of no more than MAXSENDLEN bytes. There is no limit if this parameter is not specified. |

# DM_REMOTE_DOMAINS Section

This section identifies the known set of remote domains and their characteristics.

The format of the DM_REMOTE_DOMAINS section of the DMCONFIG file is illustrated in the following listing.

**Listing 5-11 Syntax for DM_REMOTE_DOMAINS Section of DMCONFIG**

```
RDOM   required parameters [optional parameters]
```

*RDOM* is an *identifier* value used to identify each remote domain known to this configuration and must be unique within the configuration.

The following table describes the parameters that are set in the DM_REMOTE_DOMAINS section.

**Table 5-8 Parameters for the DM_REMOTE_DOMAINS section**

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| TYPE = *identifier* | Required | None | Is used for grouping remote domain into classes. TYPE can be set to TDOMAIN or any other domain gateway type. 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. For use with TMA TCP Gateway, specify TYPE=IDOMAIN. |

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| DOMAINID = *string* | Required | None | 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 31 characters or fewer. 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". |
| CODEPAGE = *table-identifier* | Optional | None | 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. The name of the file, located in the $TUXDIR/udatajobj/codepage directory, is a composite of the code page numbers used for the translation.<br><br>For example, specifying 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. For complete character listings, refer to "Code Page Translation Tables." |

# DM_ACCESS_CONTROL Section

This section is optional in the DMCONFIG file and specifies the access control lists used by local domain.

The format of the DM_ACCESS_CONTROL section of the DMCONFIG file is illustrated in the following listing.

**Listing 5-12 Syntax for DM_ACCESS_CONTROL Section of DMCONFIG**

```
ACL_NAME    required parameters
```

*ACL_NAME* is a name (*identifier*) used to identify a particular access control list; it must be 15 characters or less in length.

**Table 5-9 Parameters for DM_ACCESS_CONTROL Section**

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| ACLIST = *identifier* [,*identifier*] | Required | None | 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. |

# 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.

The format of the DM_LOCAL_SERVICES section of the DMCONFIG file is illustrated in the following listing.

**Listing 5-13 Syntax for DM_LOCAL_SERVICES Section of DMCONFIG File**

```
service     [optional parameters]
```

*service* is the local name (*identifier*) of the exported service, and it must be 15 characters or fewer 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.

**Table 5-10 Parameters for DM_LOCAL_SERVICES section**

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| ACL = *identifier* | Optional | None | 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 is not performed for requests to this service. |
| LDOM = *identifier* | Optional | None | Specifies the name identifying the local domain exporting this service. If this keyword is not specified, then all the local domains defined in the DM_LOCAL_DOMAINS section accept requests to this local service. |
| RNAME = *string* | Optional | The service name in the GWICONFIG | Specifies the name exported to remote domains. The remote domains use this name for request to this service. If this parameter is not specified, the local service name is supposed to be the name used by any remote domain. For TMA TCP, this should match the service name in the GWICONFIG. |

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| INBUFTYPE= "*type:subtype* \| FML:*servicename*" | Optional | None | Specifies the Tuxedo buffer type for replies from local servers.<br>To use an alternate data mapping product, specify FML as the buffer type (regardless of the actual Tuxedo buffer type being passed) and the encoding service name as the subtype. Because the gateway does not do the encoding, the FML buffer type in the DMCONFIG file is a flag for alternate data mapping only. The Tuxedo server and the alternate data mapping product should be configured with the actual Tuxedo buffer type. |
| OUTBUFTYPE= "*type:subtype* \| FML:*servicename*" | Optional | None | Specifies the Tuxedo buffer type for requests to local servers.<br>To use an alternate data mapping product, specify FML as the buffer type (regardless of the actual Tuxedo buffer type being passed) and the decoding service name as the subtype. Because the gateway does not do the decoding, the FML buffer type in the DMCONFIG file is a flag for alternate data mapping only. The Tuxedo server and the alternate data mapping product should be configured with the actual Tuxedo buffer type. |

# DM_REMOTE_SERVICES Section

This section provides information on services "imported" and available on remote domains.

The format of the DM_REMOTE_SERVICES section of the DMCONFIG file is illustrated in the following listing.

**Listing 5-14 Syntax for DM_REMOTE_SERVICES Section of DMCONFIG**

```
service     [optional parameters]
```

*service* is the name (*identifier*) used by the local Tuxedo System/Domain application for a particular remote service. Remote services are associated with a particular remote domain.

**Table 5-11 Parameters for the DM_REMOTE_SERVICES section**

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| CONV = {Y \| N} | Optional | N | Specifies whether the remote service is a conversational service. Use Y to specify the remote service is a conversational service. Use N to specify the remote service is not a conversational service. |
| LDOM = *identifier* | Optional | None | 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 are able to accept requests to this remote service. The service request is redirected to a remote domain of the same type (see the following definition for RDOM keyword). |
| RDOM = *identifier* | Optional | None | 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 the following definition for 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. |

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| RNAME = *string* | Optional | The service name in the GWICONFIG | 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*. For TMA TCP, this name should match the service name in the GWICONFIG file. |
| ROUTING = *identifier* | Optional | None | 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* | Optional | 30 seconds | 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. A value of 0 implies the maximum time-out value for the machine. |

| Parameter | Required/Optional | Default | Description |
|---|---|---|---|
| INBUFTYPE="*type:subtype* \| FML:*servicename*" | Optional | None | Specifies the Tuxedo buffer type for requests from local clients.<br>To use an alternate data mapping product, specify FML as the buffer type (regardless of the actual Tuxedo buffer type being passed) and the encoding service name as the subtype. Because the gateway does not do the encoding, the FML buffer type in the DMCONFIG file is a flag for alternate data mapping only. The Tuxedo application and the alternate data mapping product should be configured with the actual Tuxedo buffer type. |
| OUTBUFTYPE="*type:subtype* \| FML:*servicename*" | Optional | None | Specifies the Tuxedo buffer type for replies to local clients<br>To use an alternate data mapping product, specify FML as the buffer type (regardless of the actual Tuxedo buffer type being passed) and the decoding service name as the subtype. Because the gateway does not do the decoding, the FML buffer type in the DMCONFIG file is a flag for alternate data mapping only. The Tuxedo application and the alternate data mapping product should be configured with the actual Tuxedo buffer type. |

# DM_ROUTING Section

This section is optional in the DMCONFIG file and provides information for data-dependent routing of service requests using FML, VIEW, X_C_TYPE, and X_COMMON typed buffers.

The format of the DM_ROUTING section of the DMCONFIG file is illustrated in the following listing.

**Listing 5-15 Syntax for DM_ROUTING Section of DMCONFIG File**

```
CRITERION_NAME    required parameters
```

*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.

**Table 5-12 Parameters for DM_ROUTING Section**

| Parameter | Required /Optional | Default | Description |
|---|---|---|---|
| FIELD = *identifier* | Required | None | 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* | Required | None | 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.<br><br>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’).<br><br>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.<br><br>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. |

| Parameter | Required /Optional | Default | Description |
|---|---|---|---|
| | | | 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 ‘‘*’’ (wildcard) in the position of a range indicates any values not covered by the other ranges previously seen in the entry; only one wildcard range is allowed per entry and it should be last (ranges following it are ignored).<br><br>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.<br><br>String range values for string, array, and character field types must be placed inside a pair of single quotes and cannot 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.<br><br>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.<br><br>Within a range/RDOM pair, the range is separated from the RDOM by a ‘‘:’’. |

| Parameter | Required /Optional | Default | Description |
|---|---|---|---|
| BUFTYPE = ~*type1*[*:subtype1[,subtype2* . . . ]][;*type2[:subtype3*[, . . . ]]] . . .~ | Required | None | 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 cannot 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.<br>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.<br>If the field value is not set (for FML buffers), or does not match any specific range and a wildcard range has not been specified, an error is returned to the application process that requested the execution of the remote service. |

# Sample DMCONFIG File

Listing 5-16 is a sample DMCONFIG file and must be set up prior to running the TMA TCP Gateway product. For a sample DMCONFIG file that uses alternate data mapping tools, refer to the "Configuring BEA TMA TCP Gateway for Data Mapping" section.

**Listing 5-16 Sample DMCONFIG File**

```
#
#       Copyright (c) 1996 BEA Systems, Inc
#       All Rights Reserved
#
#       THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF
#       BEA Systems, Inc.
#       The copyright notice above does not evidence any
#       actual or intended publication of such source code.
#
#
```

```
*DM_LOCAL_DOMAINS
LOCAL          GWGRP=GROUP
               TYPE=IDOMAIN
               DOMAINID="LOCAL"

#
*DM_REMOTE_DOMAINS
REMOTE         TYPE=IDOMAIN
               DOMAINID="REMOTE"

#
*DM_LOCAL_SERVICES
ECHOX          RNAME="ZECHOSV4"
TOLOWER        RNAME="TUXTOLOWER"
               INBUFTYPE=string
               OUTBUFTYPE=string

#
*DM_REMOTE_SERVICES
TOUPPER        RDOM=REMOTE
               LDOM=LOCAL
               RNAME="TUXTOUPPER"


ECHOFAR        RDOM=REMOTE
               LDOM=LOCAL
               RNAME="BEASVR07"
               INBUFTYPE="VIEW:myview"
               OUTBUFTYPE="FML"

NORMAL         RDOM=REMOTE
               LDOM=LOCAL
               RNAME="TST1V"
```

CHAPTER 6

# Starting BEA TMA TCP Gateway

The following topics provide information about starting the BEA Tuxedo Mainframe Adapter for TCP Gateway (hereafter referenced as TMA TCP Gateway) product:

- Setting Environment Variables
- Invoking TMA TCP Gateway
- Administering the Gateways

## Setting Environment Variables

Before you attempt to use TMA TCP Gateway, you must set the TUXDIR and PATH environment variables as the following example illustrates.

**Listing 6-1 Setting ROOTDIR and PATH Environment Variables**

```
TUXDIR=/usr/tuxedo; export TUXDIR
PATH=$PATH:$TUXDIR/bin; export PATH
GWICONFIG=$APPDIR/gwiconfig;export GWICONFIG
BDMCONFIG=$APPDIR/bdmconfig;export BDMCONFIG
TUXCONFIG=$APPDIR/tuxconfig;export TUXCONFIG
```

You should set TUXDIR to the actual path where your BEA Tuxedo and TMA TCP Gateway software is installed. Set APPDIR to the application directory, similar to the UBBCONFIG file. You may also need to set the LANG environment variable if you have generated custom mapping tables or message catalogs. Some platforms may require that LANG always is set. Consult your operating system documentation for the appropriate LANG value.

# Invoking TMA TCP Gateway

Perform a tmloadcf and dmloadcf to load the UBBCONFIG and the DMCONFIG files. Invoke TMA TCP Gateway using the tmboot command to boot the BEA Tuxedo system. Be sure to configure TMA TCP Gateway prior to using the tmboot command. For details, see the "Configuring BEA TMA TCP Gateway" section.

**Listing 6-2 Invoking TMA TCP Gateway Using tmboot**

```
tmboot
```

# Administering the Gateways

BEA Tuxedo has a set of tmadmin and dmadmin commands for the administration of the TMA TCP Gateways. For detailed information about these commands, refer to the *BEA Tuxedo Administrator's Guide* and the *BEA Tuxedo Domain User Guide*.

APPENDIX A

# Error and Information Messages

There are several ways that local client programs can learn about application errors that occur on remote systems. For example:

- Application failures can be communicated through error indicators that remote systems send to the local Tuxedo Mainframe Adapter for TCP Gateway.
- Service routines can include information about failures in output records that are returned to client programs.

This document contains a description of error and informational messages that can be encountered while using BEA Tuxedo Mainframe Adapter for TCP Gateway (hereafter referenced as TMA TCP Gateway).

| | | |
|---|---|---|
| **1000:ERROR** | **Memory allocation error** | |
| | **DESCRIPTION** | An attempt to allocate memory failed. |
| | **ACTION** | Check available memory. |
| **1001:ERROR** | **Cannot load shared module <text>: <text>** | |
| | **DESCRIPTION** | An attempt to load the listed shared library failed. The reason is noted in the error text. |
| | **ACTION** | Verify that the shared library exists. |

| | | |
|---|---|---|
| **1002:ERROR** | **Cannot get symbol '<text>' in module '<text>'** | |
| | **DESCRIPTION** | The listed symbol was not found in the listed shared module. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1003:ERROR** | **No encoding type defined for <text>. No encoding done** | |
| | **DESCRIPTION** | The gateway configuration file does not specify a `TYPE` parameter for the listed `FOREIGN` gateway or a `DFLTTYPE` parameter in the `GLOBAL` section. |
| | **ACTION** | Correct the gateway configuration file. |
| **1004:ERROR** | **Encoding type '<text>:<text>' for type '<text>' failed.** | |
| | **DESCRIPTION** | Allocating a typed buffer for encoding output failed. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1005:ERROR** | **Decode failed due to allocation error** | |
| | **DESCRIPTION** | The output data buffer was not created correctly with `tpalloc`( ). |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1006:ERROR** | **No encoding type defined for <text>. No decoding done** | |
| | **DESCRIPTION** | The Domain Gateway library software does not have an encoding type defined in shared memory for the listed gateway. |
| | **ACTION** | Verify the `TYPE` parameter for the gateway in the gateway configuration file. |

| | | |
|---|---|---|
| **1007:ERROR** | **Decoding type <text>:<text> for type <text> failed** | |
| | **DESCRIPTION** | The Domain Gateway library software detected an error while getting decoding information for the listed type, subtype, and remote type. |
| | **ACTION** | Previous logged error messages should provide the exact reasons for the failure. Contact your Tuxedo System Technical Support. |
| **1008:ERROR** | **GWICONFIG env var not found!** | |
| | **DESCRIPTION** | Retrieving the environment variable `GWICONFIG` failed. |
| | **ACTION** | Set and export the `GWICONFIG` environment variable. |
| **1009:ERROR** | **<text> config file not found!** | |
| | **DESCRIPTION** | The named configuration file cannot be found. |
| | **ACTION** | Verify that the file which the `GWICONFIG` environment variable points to exists. |
| **1010:ERROR** | **Parse of <text> failed, exiting** | |
| | **DESCRIPTION** | An attempt to parse the gateway configuration file failed. |
| | **ACTION** | Verify the format of the gateway configuration file. |
| **1011:ERROR** | **<number> errors found while parsing <text>, exiting** | |
| | **DESCRIPTION** | Errors occurred while parsing the gateway configuration file. |
| | **ACTION** | Verify the format of the gateway configuration file |

| | | |
|---|---|---|
| **1012:INFO** | **Dumping configuration to stdout** | |
| | **DESCRIPTION** | The current configuration is dumped to the standard output. |
| | **ACTION** | No action required. |
| **1013:ERROR** | **Sending error to <text>:\n\t<text>** | |
| | **DESCRIPTION** | The error condition described is being sent to the listed gateway. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1014:ERROR** | **Invalid action detected <number>** | |
| | **DESCRIPTION** | The Domain Gateway library software detected an invalid action while trying to encode a buffer. |
| | **ACTION** | This is an internal error with no associated user action. If the error persists, contact your Tuxedo System Technical Support with the exact error message. |
| **1015:ERROR** | **Data encoding failed** | |
| | **DESCRIPTION** | The Domain Gateway library software detected an error encoding data. |
| | **ACTION** | Previous logged error messages should provide the exact reasons for the failure. Contact your Tuxedo System Technical Support. |
| **1016:ERROR** | **Invalid wrap switch index** | |
| | **DESCRIPTION** | The wrap switch index in shared memory is not valid. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |

| | | |
|---|---|---|
| **1017:ERROR** | **Buffer wrapping failed** | |
| | **DESCRIPTION** | The buffer wrapping function returned an error. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1018:ERROR** | **Bad wrapping protocol** | |
| | **DESCRIPTION** | The wrapping protocol in shared memory is not `GWI_PROTOCOL`. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1019:ERROR** | **Cannot unwrap buffer** | |
| | **DESCRIPTION** | The buffer unwrapping function returned an error. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1020:ERROR** | **Unable to obtain local service (<text>) information from shared memory** | |
| | **DESCRIPTION** | Information about the local service could not be found in shared memory. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1021:ERROR** | **Request for unknown service '<text>' from <text>** | |
| | **DESCRIPTION** | Information about the listed service could not be found in shared memory. |
| | **ACTION** | Verify the `DMCONFIG` file and ensure `dmloadcf` has been run on it. |
| **1022:ERROR** | **Reply received for invalid session** | |
| | **DESCRIPTION** | The Domain Gateway received a reply for an invalid session. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |

| | | |
|---|---|---|
| **1023:ERROR** | **Reply received for inactive session** | |
| | **DESCRIPTION** | The Domain Gateway received a reply for an inactive session. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1024:ERROR** | **Unable to obtain remote service (<text>) information from shared memory** | |
| | **DESCRIPTION** | Information about the remote service could not be found in shared memory. |
| | **ACTION** | Verify the `DMCONFIG` file and insure `dmloadcf` has been run on it. |
| **1025:ERROR** | **Data decoding failed** | |
| | **DESCRIPTION** | The Domain Gateway library software detected an error while decoding a received message buffer. |
| | **ACTION** | Previous logged error messages should provide the exact reasons for the failure. Contact your Tuxedo System Technical Support. |
| **1026:INFO** | **<text> : <text>** | |
| | **DESCRIPTION** | The field name and its value in a fielded buffer are logged. |
| | **ACTION** | No action required. |
| **1027:INFO** | **Type <number> : <text>** | |
| | **DESCRIPTION** | The field type and its value in a fielded buffer are logged. |
| | **ACTION** | No action required. |
| **1028:ERROR** | **Cannot open <text>:<text>** | |
| | **DESCRIPTION** | Could not find the listed view file. |
| | **ACTION** | Verify the `VIEWDIR` and `VIEWFILES` environment variables. |

| | | |
|---|---|---|
| **1029:ERROR** | **FML incompatible field in <text>:<text>** | |
| | **DESCRIPTION** | The listed view contains the packed decimal data type. Views containing this type cannot be used for FML-to-view or view-to-view conversions. |
| | **ACTION** | Remove packed decimal fields from the input FML or view, or send the same view structure to the local gateway that is sent to the remote gateway. |
| **1030:ERROR** | **tpalloc for <text> failed** | |
| | **DESCRIPTION** | Allocation of an Tuxedo typed buffer failed. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1031:ERROR** | **Fvstof failed for <text>:<text>** | |
| | **DESCRIPTION** | The conversion of a C structure to a fielded buffer failed. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1032:INFO** | **<text>, licensed to <text>** | |
| | **DESCRIPTION** | Informational message describing license information. |
| | **ACTION** | None required. |
| **1033:ERROR** | **tpalloc for <text>:<text> failed** | |
| | **DESCRIPTION** | Allocating a typed buffer for the listed view failed. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1036:ERROR** | **Unable to obtain network descriptor; error code = <number>** | |
| | **DESCRIPTION** | This is an internal error code, indicating a network initialization failure. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |

| | | |
|---|---|---|
| **1037:ERROR** | **Unable to determine RDOM for request** | |
| | **DESCRIPTION** | This is an internal error code, indicating a shared memory lookup failure. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1038:ERROR** | **Unable to connect to <text>; return code <number>** | |
| | **DESCRIPTION** | The gateway detected an error while trying to open a connection on a socket. |
| | **ACTION** | Verify that the network address information for the desired `FOREIGN` entry is correct. |
| **1039:ERROR** | **Connection failed to <text>** | |
| | **DESCRIPTION** | The Domain Gateway library software detected an error while trying to connect to the listed gateway. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1048:ERROR** | **Unable to send service request** | |
| | **DESCRIPTION** | The Domain Gateway library software detected an error while trying to send a message to the network. |
| | **ACTION** | This is an internal error with no associated user action. If the error persists, contact your Tuxedo System Technical Support with the exact error message. |
| **1049:ERROR** | **Unable to send service request \"<text>\"** | |
| | **DESCRIPTION** | The Domain Gateway library software detected an error while trying to send a message to the network. |
| | **ACTION** | This is an internal error with no associated user action. If the error persists, contact your Tuxedo System Technical Support with the exact error message. |

| | | |
|---|---|---|
| **1050:ERROR** | **Unable to get remote service information from shared memory** | |
| | **DESCRIPTION** | Information about the remote service could not be found in shared memory. |
| | **ACTION** | Verify the `DMCONFIG` file and insure `dmloadcf` has been run on it. |
| **1052:ERROR** | **Can't get remote service name for <text>** | |
| | **DESCRIPTION** | The remote service name for the listed service could not be found in shared memory. |
| | **ACTION** | Verify the `DMCONFIG` file and insure `dmloadcf` has been run on it. |
| **1053:ERROR** | **No remote service entry for <text>** | |
| | **DESCRIPTION** | No remote service entry for the listed service was found in shared memory. |
| | **ACTION** | Correct the gateway configuration file. |
| **1054:ERROR** | **Cannot suspend action** | |
| | **DESCRIPTION** | The Domain Gateway library software was unable to suspend the action while trying to set up a network connection. |
| | **ACTION** | This is an internal error with no associated user action. If the error persists, contact your Tuxedo System Technical Support with the exact error message. |
| **1055:WARN** | **CICS init string failed for <text>.** | |
| | **DESCRIPTION** | The Domain Gateway library software detected an error while trying to send a CICS initialization string to the listed gateway. |
| | **ACTION** | This is an internal error with no associated user action. If the error persists, contact your Tuxedo System Technical Support with the exact error message. |

| | | |
|---|---|---|
| **1056:INFO** | **Connection <number> initialized to <text>, ID:\n\t<text>** | |
| | **DESCRIPTION** | The Domain Gateway library software successfully initialized a connection to the gateway. |
| | **ACTION** | No action required. |
| **1057:INFO** | **Multiplex count for connection <number> is <number>** | |
| | **DESCRIPTION** | The multiplex count for the listed connection is given. |
| | **ACTION** | No action required. |
| **1058:INFO** | **Listen port <number> (<text>) established** | |
| | **DESCRIPTION** | The listen port for the gateway has been established. |
| | **ACTION** | No action required. |
| **1059:ERROR** | **Cannot change action into msg_failure** | |
| | **DESCRIPTION** | The action cannot be changed to `msg_failure`. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1060:ERROR** | **Cannot change action into msg_reply** | |
| | **DESCRIPTION** | The action cannot be changed to `msg_failure`. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1062:ERROR** | **Unexpected action state <number>** | |
| | **DESCRIPTION** | This message should never occur. The program code or defines are wrong if it does. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |

| | | |
|---|---|---|
| **1063:WARN** | **Conversation not found - closing connection** | |
| | **DESCRIPTION** | The ID of the connection was not found in the shared memory conversation table. |
| | **ACTION** | No action required. |
| **1064:ERROR** | **No more actions** | |
| | **DESCRIPTION** | Trying to create an action to time out a conversation failed. |
| | **ACTION** | No action required. |
| **1066:ERROR** | **Cannot allocate appropriate array for poll** | |
| | **DESCRIPTION** | The Domain Gateway library software was unable to allocate the `pollfd` array. |
| | **ACTION** | This is an internal error with no associated user action. If the error persists, contact your Tuxedo System Technical Support with the exact error message. |
| **1067:ERROR** | **Run-time environment setting failure** | |
| | **DESCRIPTION** | The Domain Gateway library software was unable to initialize the run-time environment. |
| | **ACTION** | Previous logged error messages should provide the exact reasons for the failure. Contact your Tuxedo System Technical Support. |
| **1068:ERROR** | **No free listen structures available** | |
| | **DESCRIPTION** | Obtaining a free listen structure failed. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |

| | | |
|---|---|---|
| **1069:ERROR** | **Cannot listen for host <text>** | |
| | **DESCRIPTION** | The Domain Gateway library software was unable to resolve the IP address of the listed gateway. |
| | **ACTION** | Previous logged error messages should provide the exact reasons for the failure. Contact your Tuxedo System Technical Support. |
| **1070:ERROR** | **Gateway fails to complete connection** | |
| | **DESCRIPTION** | The Domain Gateway library software was unable to open a socket connection. |
| | **ACTION** | Verify that the address of the remote host is configured correctly. |
| **1071:ERROR** | **Unable to establish listen port <number> (<text>)** | |
| | **DESCRIPTION** | The Domain Gateway library software was unable to establish a listen socket. |
| | **ACTION** | Verify that the configured listen port is not in use by another process. Contact your Tuxedo System Technical Support. |
| **1072:ERROR** | **Can't read CODEPAGE '<text>' for <text> DOMAIN <text>** | |
| | **DESCRIPTION** | The `codepage` file specified in the `DMCONFIG` file cannot be opened. |
| | **ACTION** | Verify that the specified file is available and that the file protections are set correctly. |
| **1073:ERROR** | **<text> '<text>' not found in domain configuration** | |
| | **DESCRIPTION** | A `FOREIGN` or `NATIVE` entry does not have a corresponding entry in the domain configuration (`dmconfig`) file. |
| | **ACTION** | Verify that each `FOREIGN` and `NATIVE` entry in the `gwiconfig` file has a corresponding entry in the `dmconfig` file. |

| | | |
|---|---|---|
| **1074:ERROR** | **Unlicensed Tuxedo System Binary** | |
| | **DESCRIPTION** | The files in your Tuxedo System installation do not contain the expected software license information for TMA TCP. |
| | **ACTION** | Reinstall the TMA TCP System software, using the license token and serial number supplied with the distribution media. Make sure not to terminate the installation program prematurely, because the license information is processed at the end. |
| **1075:ERROR** | **Cannot alloc conv switch** | |
| | **DESCRIPTION** | Creating a conversation switch in shared memory failed. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1076:ERROR** | **Cannot alloc wrap switch** | |
| | **DESCRIPTION** | Creating a wrap switch in shared memory failed. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1077:ERROR** | **Couldn't load configuration** | |
| | **DESCRIPTION** | The Domain Gateway library software was unable to load the configuration. |
| | **ACTION** | Verify the format of the gateway configuration file. |
| **1078:ERROR** | **No gateway defined in config file** | |
| | **DESCRIPTION** | The gateway configuration file does not contain a gateway definition. |
| | **ACTION** | Correct the gateway configuration file. |

| | | |
|---|---|---|
| **1079:ERROR** | **Couldn't find <text> gateway (GWINAME) in NATIVE section** | |
| | **DESCRIPTION** | The listed gateway was not found in the `NATIVE` section of the gateway configuration file. |
| | **ACTION** | Correct the gateway configuration file. |
| **1080:ERROR** | **Couldn't find env var GWINAME. Cannot guess which gateway.** | |
| | **DESCRIPTION** | The gateway configuration file contains more than one gateway entry and the environment variable `GWINAME` is not set. |
| | **ACTION** | Set the `GWINAME` environment variable. |
| **1081:ERROR** | **Couldn't find host name. Cannot guess gateway address.** | |
| | **DESCRIPTION** | The Domain Gateway library software detected an error while trying to get the name of the current host. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1082:ERROR** | **Service <text> is SECURE, Tuxedo security disabled** | |
| | **DESCRIPTION** | The listed service requires security, but Tuxedo security is disabled. |
| | **ACTION** | Either turn on Tuxedo security with the `SECURITY` resource, or remove the `SECURE` parameter from the gateway configuration file for this service. |
| **1083:ERROR** | **Network initialization fails; error code = <number>** | |
| | **DESCRIPTION** | Initialization of the network protocol has failed. |
| | **ACTION** | Verify that the underlying TCP product is installed and configured correctly. |

| | | |
|---|---|---|
| **1084:ERROR** | **Universal callback setup fails; error code = <number>** | |
| | **DESCRIPTION** | This is an internal error code, indicating a library setup failure. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1085:ERROR** | **Unable to register protocol; error code = <number>** | |
| | **DESCRIPTION** | This is an internal error code, indicating a library setup failure. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1086:ERROR** | **Unable to set network device; error code = <number>** | |
| | **DESCRIPTION** | The configured network device cannot be found. |
| | **ACTION** | Verify that the `NWDEVICE` field in the `LOCAL` section of the `GWICONFIG` file is correct (usually /dev/tcp). |
| **1087:ERROR** | **Can't get remote address** | |
| | **DESCRIPTION** | Could not convert the remote internet address to a binary address. |
| | **ACTION** | Previous logged error messages should provide the exact reasons for the failure. Contact your Tuxedo System Technical Support. |
| **1088:INFO** | **Accepted new connection <number> from <text>** | |
| | **DESCRIPTION** | A new connection from the remote gateway has been accepted. |
| | **ACTION** | No action required. |

| | | |
|---|---|---|
| **1089:ERROR** | **Poll returned error (num_cnx=<number>)** | |
| | **DESCRIPTION** | The Domain Gateway library software detected an error while polling the connection for events. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1090:ERROR** | **No free sessions available** | |
| | **DESCRIPTION** | The Domain Gateway library software was unable to allocate shared memory for a new session structure. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1091:ERROR** | **Unable to get new action: gpnd <number>** | |
| | **DESCRIPTION** | This is an internal error, indicating that the Domain Gateway is unable to allocate a new action. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1092:ERROR** | **Invalid ses_idx <number> for act <number>** | |
| | **DESCRIPTION** | The action structure references an invalid session index. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1093:ERROR** | **Unable to allocate a receive buffer** | |
| | **DESCRIPTION** | The Domain Gateway is unable to allocate memory for a receive buffer. |
| | **ACTION** | Verify that system memory requirements are met. If problems persist, contact your Tuxedo System Technical Support. |

| | | |
|---|---|---|
| **1094:WARN** | **Deleting late reply from local service '<text>'** | |
| | **DESCRIPTION** | A late reply has arrived from a Tuxedo service. Because the request has already timed out, the reply is discarded. |
| | **ACTION** | No action is required. If an excessive number of timeouts occur, adjustments may be needed in the configuration. |
| **1097:ERROR** | **Error message from <text>:\n\tTuxedo code <number> (<text>):\n\t<text>** | |
| | **DESCRIPTION** | Received an error message from the remote gateway. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1098:WARN** | **Late response received for freed session <number>/<number>** | |
| | **DESCRIPTION** | Received a response on a file descriptor which had timed out. |
| | **ACTION** | No action required. |
| **1099:INFO** | **Stale message found** | |
| | **DESCRIPTION** | Received a message on a session with no associated action. |
| | **ACTION** | No action required. |
| **1100:ERROR** | **Data conversion failed for local request '<text>'** | |
| | **DESCRIPTION** | The Domain Gateway library software detected an error while trying to decode a message. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1101:ERROR** | **Unexpected diagnostic <number> returned from server** | |
| | **DESCRIPTION** | An unexpected diagnostic flag was returned from the server. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |

| | | |
|---|---|---|
| **1102:ERROR** | **Data conversion failed for local service '<text>'** | |
| | **DESCRIPTION** | The Domain Gateway library software detected an error while trying to encode a message. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1103:INFO** | **Connection <number> initialized from <text>** | |
| | **DESCRIPTION** | The Domain Gateway library software received a connection request from the remote gateway. |
| | **ACTION** | No action required |
| **1104:ERROR** | **Unable to identify local service, rname '<text>'** | |
| | **DESCRIPTION** | The service requested by the listed remote service was not found in shared memory. |
| | **ACTION** | Verify the `DMCONFIG` file and insure `dmloadcf` has been run on it. |
| **1105:ERROR** | **Unable to map local service RNAME <text>** | |
| | **DESCRIPTION** | The service requested by the listed service name was not found in shared memory. |
| | **ACTION** | Verify the `DMCONFIG` file and insure `dmloadcf` has been run on it. |
| **1106:ERROR** | **Unable to identify user <text>** | |
| | **DESCRIPTION** | The listed user is not authorized for the requested service. |
| | **ACTION** | Verify the `tpusr` file in `APPDIR`. |
| **1107:ERROR** | **Protocol error: server cannot use GWI_OP_DISCON** | |
| | **DESCRIPTION** | A remote service is sending a disconnect. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |

| | | |
|---|---|---|
| **1108:ERROR** | **Invalid opcode on receive** | |
| | **DESCRIPTION** | The Domain Gateway library software received an invalid opcode. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1110:INFO** | **Closing connection <number>** | |
| | **DESCRIPTION** | The listed connection is being closed. |
| | **ACTION** | No action required. |
| **1111:ERROR** | **gwi_mk_error called with bad act_idx!** | |
| | **DESCRIPTION** | The Domain Gateway library software detected that an error occurred between a send and receive. The connection closes. |
| | **ACTION** | No action required. |
| **1112:ERROR** | **Too many unrecoverable errors occurred - deleting action** | |
| | **DESCRIPTION** | The Domain Gateway library software detected more than 2 errors while trying to complete the action. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1113:ERROR** | **Unrecoverable error occurred on send of reply request - deleting action** | |
| | **DESCRIPTION** | Cannot send a reply to a call made with `TPNOREPLY`. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1114:ERROR** | **Local request '<text>' timed out, returning error response** | |
| | **DESCRIPTION** | The request for the local service timed out. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |

| | | |
|---|---|---|
| **1115:WARN** | **Deleting timed out session <number>/<number>** | |
| | **DESCRIPTION** | The request for the local service timed out. |
| | **ACTION** | No action required. |
| **1116:ERROR** | **Local request '<text>' failed, returning error response** | |
| | **DESCRIPTION** | The request for the local service failed. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1117:ERROR** | **Error on receive from <text>** | |
| | **DESCRIPTION** | The Domain Gateway library software detected an error when receiving a request from a remote gateway. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1118:ERROR** | **Unrecoverable error occurred on unknown receipt(2) - deleting action** | |
| | **DESCRIPTION** | The current action has no associated session. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1119:ERROR** | **No FOREIGN entry for REMOTE_DOMAIN '<text>'** | |
| | **DESCRIPTION** | There is no `FOREIGN` entry in the `GWICONFIG` file corresponding to the specified `REMOTE_DOMAIN`. |
| | **ACTION** | Create a `FOREIGN` entry in the `GWICONFIG` file for each `REMOTE_DOMAIN` that is available through the gateway. |
| **1120:ERROR** | **Unrecoverable error occurred - deleting GWEV_MSG_FAILURE action (type <number>)** | |
| | **DESCRIPTION** | The Domain Gateway library software detected an unrecoverable error. |
| | **ACTION** | Previous logged error messages should provide more information. Contact your Tuxedo System Technical Support. |

| | | |
|---|---|---|
| **1121:ERROR** | **Unrecoverable error occurred - changing action (type <number>) into GWEV_MSG_FAILURE** | |
| | **DESCRIPTION** | The Domain Gateway library software detected an unrecoverable error. |
| | **ACTION** | Previous logged error messages should provide more information. Contact your Tuxedo System Technical Support. |
| **1122:ERROR** | **Unrecoverable error occurred - deleting action (type <number>)** | |
| | **DESCRIPTION** | The Domain Gateway library software detected an unrecoverable error. |
| | **ACTION** | Previous logged error messages should provide more information. Contact your Tuxedo System Technical Support. |
| **1123:INFO** | **Request id not found** | |
| | **DESCRIPTION** | The action with the given request ID was not found in shared memory. |
| | **ACTION** | No action required. |
| **1124:ERROR** | **convid not found** | |
| | **DESCRIPTION** | The session with the given conversation ID was not found in shared memory. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1125:ERROR** | **Invalid network address <<text>>** | |
| | **DESCRIPTION** | The given network address is not valid. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1126:ERROR** | **Invalid host name <<text>>** | |
| | **DESCRIPTION** | Could not get the network address for the listed host. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |

| | | |
|---|---|---|
| **1127:ERROR** | **Bad ip adress format <<text>>** | |
| | **DESCRIPTION** | The format of the listed ip address is invalid. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1128:ERROR** | **Input string too long <<text>>** | |
| | **DESCRIPTION** | The address is too long to fit in the memory allocated for it. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1129:ERROR** | **Timeout on remote service <text> on <text>** | |
| | **DESCRIPTION** | The Domain Gateway library software detected a timeout on the listed remote service. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1130:ERROR** | **Timeout on local service <text>, returning error to <text>** | |
| | **DESCRIPTION** | The Domain Gateway library software detected a timeout on the listed local service. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1131:ERROR** | **Connect time-out on connection <number> to <text>** | |
| | **DESCRIPTION** | The Domain Gateway library software detected a timeout while attempting to connect to the remote gateway. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1132:ERROR** | **Idle time-out on connection <number> to <text>** | |
| | **DESCRIPTION** | The Domain Gateway library software detected a timeout on the listed connection to the remote gateway. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |

| | | |
|---|---|---|
| **1133:ERROR** | **Parse error: <text> line <number> column <number> (after <<text>>) at <<text>>** | |
| | **DESCRIPTION** | The Domain Gateway library software encountered an error while parsing the gateway configuration file. |
| | **ACTION** | Correct the gateway configuration file. |
| **1134:ERROR** | **No NWDEVICE specified** | |
| | **DESCRIPTION** | There was no `NWDEVICE` specified in the gateway configuration file. |
| | **ACTION** | Correct the gateway configuration file. |
| **1135:ERROR** | **NATIVE name <text> already in use** | |
| | **DESCRIPTION** | The listed gateway name is specified more than once in the `NATIVE` section of the gateway configuration file. |
| | **ACTION** | Correct the gateway configuration file. |
| **1136:WARN** | **Unreasonable POLL_TIME <number> ignored for NATIVE <text>** | |
| | **DESCRIPTION** | The gateway configuration file specifies a poll time which is less than 100,000 or greater than 10,000,000. |
| | **ACTION** | No action required. The poll time is set to 250,000. |
| **1137:ERROR** | **REMOTE_SERVICE <text> has no OUTREQ_TIME, no default in GLOBAL** | |
| | **DESCRIPTION** | The gateway configuration file does not have an `OUTREQ_TIME` specified for a remote service. |
| | **ACTION** | Correct the gateway configuration file. |
| **1138:ERROR** | **FOREIGN name <text> already in use** | |
| | **DESCRIPTION** | The listed gateway name is specified more than once in the `FOREIGN` section of the gateway configuration file. |

| | | |
|---|---|---|
| | **ACTION** | Correct the gateway configuration file. |
| **1139:ERROR** | **Missing mandatory TYPE param for FOREIGN <text>** | |
| | **DESCRIPTION** | The gateway configuration file does not have a `TYPE` parameter for the listed gateway. |
| | **ACTION** | Correct the gateway configuration file. |
| **1140:ERROR** | **Missing mandatory WRAP param for FOREIGN <text>** | |
| | **DESCRIPTION** | The gateway configuration file does not have a `WRAP` parameter for the listed gateway. |
| | **ACTION** | Correct the gateway configuration file. |
| **1141:ERROR** | **Missing mandatory IP_ADDR param for FOREIGN <text>** | |
| | **DESCRIPTION** | The gateway configuration file does not have an `IP_ADDR` parameter for the listed gateway. |
| | **ACTION** | Correct the gateway configuration file. |
| **1142:ERROR** | **LOCAL_SERVICE name <text> already in use** | |
| | **DESCRIPTION** | The listed service name is specified more than once in the `LOCAL_SERVICE` section of the gateway configuration file. |
| | **ACTION** | Correct the gateway configuration file. |
| **1145:ERROR** | **REMOTE_SERVICE name <text> already in use** | |
| | **DESCRIPTION** | The listed service name is specified more than once in the `REMOTE_SERVICE` section of the gateway configuration file. |
| | **ACTION** | Correct the gateway configuration file. |
| **1149:ERROR** | **Mandatory <text> section missing** | |
| | **DESCRIPTION** | The gateway configuration file must contain the `GLOBAL`, `FOREIGN`, `NATIVE`, `LOCAL_SERVICES`, and `REMOTE_SERVICES` sections. |
| | **ACTION** | Correct the gateway configuration file. |

| | | |
|---|---|---|
| **1150:ERROR** | **Duplicate <text> section** | |
| | **DESCRIPTION** | The gateway configuration file contains multiple entries for the listed section. |
| | **ACTION** | Correct the gateway configuration file. |
| **1151:ERROR** | **Section <text> out of order** | |
| | **DESCRIPTION** | The listed section is not in the proper order in the gateway configuration file. |
| | **ACTION** | Correct the gateway configuration file. |
| **1152:INFO** | **Section order must be *GLOBAL then *NATIVE/*FOREIGN then *LOCAL_SERVICES/*REMOTE_SERVICES** | |
| | **DESCRIPTION** | The gateway configuration file must have the following order of sections: `GLOBAL`, `NATIVE/FOREIGN`, `LOCAL_SERVICES/REMOTE_SERVICES`. |
| | **ACTION** | No action required. |
| **1153:WARN** | **Bad subtype <text> truncated to <number> characters** | |
| | **DESCRIPTION** | The gateway configuration file specifies a subtype name which is too long. |
| | **ACTION** | Correct the gateway configuration file. |
| **1154:WARN** | **Bad type <text> truncated to <number> characters** | |
| | **DESCRIPTION** | The gateway configuration file specifies a type name which is too long. |
| | **ACTION** | Correct the gateway configuration file. |
| **1155:WARN** | **SECURE= accepts Y or N, not <text>** | |
| | **DESCRIPTION** | The gateway configuration file contains an invalid entry for the `SECURE` parameter. |
| | **ACTION** | Correct the gateway configuration file. |
| **1156:** | **DSECURITY= accepts Y or N, not <parameter>** | |
| | **DESCRIPTION** | The gateway configuration file contains an invalid entry for the `DSECURITY` parameter. |

| | | |
|---|---|---|
| | **ACTION** | Correct the gateway configuration file. |
| **1157:WARN** | **DUMP= accepts Y or N, not <text>** | |
| | **DESCRIPTION** | The gateway configuration file contains an invalid entry for the `DUMP` parameter. |
| | **ACTION** | Correct the gateway configuration file. |
| **1158:WARN** | **CONNSYNC= accepts Y or N, not <text>** | |
| | **DESCRIPTION** | The gateway configuration file contains an invalid entry for the `CONNSYNC` parameter. |
| | **ACTION** | Correct the gateway configuration file. |
| **1159:WARN** | **CICS= accepts Y or N, not <text>** | |
| | **DESCRIPTION** | The gateway configuration file contains an invalid entry for the CICS parameter. |
| | **ACTION** | Correct the gateway configuration file. |
| **1160:ERROR** | **Remove obsolete line in <text> : \"REPLY = <text>\"** | |
| | **DESCRIPTION** | The gateway configuration file contains a `REPLY` parameter, which is obsolete. |
| | **ACTION** | Remove the `REPLY` parameter from the gateway configuration file. |
| **1161:ERROR** | **CONV=Y, conversational mode not supported** | |
| | **DESCRIPTION** | The gateway configuration file specifies `CONV=Y` for a local service. This is currently not supported. |
| | **ACTION** | Correct the gateway configuration file. |
| **1162:WARN** | **CONV= accepts Y or N, not <text>** | |
| | **DESCRIPTION** | The gateway configuration file contains an invalid entry for the `CONV` parameter. |
| | **ACTION** | Correct the gateway configuration file. |
| **1163:ERROR** | **Bad unwrapping protocol** | |

| | | |
|---|---|---|
| | **DESCRIPTION** | The unwrapping protocol in shared memory is not `GWI_PROTOCOL`. |
| | **ACTION** | Contact your Tuxedo System Technical Support. |
| **1164:INFO** | **External encode/decode enabled, type '<text>'** | |
| | **DESCRIPTION** | The external encoding command line option has been specified on the gateway, with the given encoding type. |
| | **ACTION** | None required. |
| **1165:ERROR** | **External encode/decode service returned error:\n\tTuxedo code <number> (<text>)** | |
| | **DESCRIPTION** | An external encode or decode request has resulted in an error. The Tuxedo code and a descriptive message are supplied. |
| | **ACTION** | Verify that the encode/decode service is running and that the `DMCONFIG` and `GWICONFIG` files are configured properly. |
| **1166:ERROR** | **TCP-IP Listener Error Received: <text>** | |
| | **DESCRIPTION** | An error has been reported by the TCP/IP listener on the remote system. |
| | **ACTION** | Examine the logs on the remote system to determine the cause of the TCP/IP failure. |
| **1167:INFO** | **Mainframe buffer length checking disabled** | |
| | **DESCRIPTION** | The gateway has been configured to allow copybook buffers from the remote host that are not as large as the desired `VIEW` buffer. |
| **1168:INFO** | **Mainframe padding disabled** | |
| | **DESCRIPTION** | The gateway has been configured to not align copybook members sent to and received from the remote host. |

| 1169:INFO | Request logging enabled | |
|---|---|---|
| | **DESCRIPTION** | The gateway has been configured to log each request to standard error output. |

# Code Page Translation Tables

This document provides the code page translation tables distributed with the BEA Tuxedo Mainframe Adapter for TCP Gateway (hereafter referenced as TMA TCP Gateway) product. The files containing these translation tables are located in `$TUXDIR/udataobj/codepage` on the product CDROM.

## 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 need 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 the "Using Code Page Translation Tables" section.

However, if you must modify a translation table, be aware of the following:

- You must have valid character mapping information. 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.
- It is not recommended that you build tables from scratch.
- 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.

**Note:** 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. With a text editor, open the file you want to modify. The following command opens the translation table file for Germany (00819x00273) in a text editor.

   ```
   edit $TUXDIR/udataobj/codepage/00819x00273
   ```

2. Using the editor functions, modify the character code in the outbound table.
3. Using the editor functions, modify the counterpart character code in the inbound table.
4. Repeat Steps 2 and 3 until you have completed the modifications.
5. Using the editor functions, save the file with a new name. Do not save the modified file using the original file name.

   **Warning:** Do *not* save modifications to any of the original files provided with your product CDROM.

6. Exit the editor.
7. 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.

# Default Tuxedo

**Listing B-1 Default Tuxedo 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)

**Listing B-2 Translation Table for United States**

```
#==================================================================
# 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
```

United States (00819x00037)

```
# 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)

**Listing B-3 Translation Table for Germany**

```
#=============================================================
# 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)

**Listing B-4 Translation Table for Sweden/Finland**

```
#================================================================
# 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)

**Listing B-5 Translation Table for Spain**

```
#=============================================================
# 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)

**Listing B-6 Translation Table for Great Britain**

```
#=================================================================
# 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)

**Listing B-7 Translation Table for France**

```
#================================================================
# 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)

**Listing B-8 Translation Table for Belgium**

```
#================================================================
# 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)

**Listing B-9 Translation Table for Portugal**

```
#=============================================================
# 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)

**Listing B-10 Latin-1 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)

**Listing B-11 Latin-2 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
```

# Index

## D



## E



## F



## G

## N



## O



## P



## R



## S

## T

## U



## V



## W



## X