BEA Tuxedo

Using the BEA Tuxedo TOP END Domain Gateway with ATMI Applications

BEA Tuxedo Release 8.0
Document Edition 8.0
June 2001

Using the BEA Tuxedo TOP END Domain Gateway with ATMI Applications

Document Edition	Date	Software Version
8.0	June 2001	BEA Tuxedo Release 8.0

Contents

About This Document

This document explains how to configure applications that use the BEA Tuxedo TOP END Domain Gateway.

This document includes the following topics:

- Chapter 1, "Introducing the BEA Tuxedo TOP END Domain Gateway," introduces the gateway and describes how it works.
- Chapter 2, "Interconnecting Systems with the TOP END Domain Gateway," discusses interoperability, connection handling, queuing handling, transaction support, application programming interfaces, and administration.
- Chapter 3, "Understanding Security Between the BEA TOP END and BEA Tuxedo Systems," describes how gateway security works.
- Chapter 4, "Configuring the TOP END Domain Gateway," describes how to configure the gateway.
- Chapter 5, "Editing the UBBCONFIG File," explains how to edit the UBBCONFIG file.
- Chapter 6, "Editing the DMCONFIG File," explains how to edit the DMCONFIG file.
- Chapter 7, "Configuring Security Between BEA TOP END and BEA Tuxedo Systems," describes how to configure security.
- Chapter 8, "Communicating with Multiple BEA TOP END Systems," describes how to configure and define multiple processes and how to modify the DMCONFIG file.
- Chapter 9, "Sample Configuration File for a TOP END Domain Gateway," describes a sample DMCONFIG file.

- Chapter 10, “Administering the TEDG During Run Time,” describes run-time administration.
- Chapter 11, “API Programming,” describes how to use the BEA Tuxedo ATMI API and the BEA TOP END CSI API.
- Chapter 12, “Request/Response Mode Programming,” describes how to use Request/Response messaging to pass messages between BEA Tuxedo clients and BEA Tuxedo TOP END clients.
- Chapter 13, “Conversational Mode Programming,” describes how to use conversational messaging.
- Chapter 14, “Reliable Queuing Programming,” describes how to use reliable queuing.
- Chapter 15, “Transactional Support Programming,” describes how to use transactions.
- Chapter 16, “Security Programming,” discusses some security programming considerations.

What You Need to Know

This document is intended primarily for application developers who are interested in writing applications that use the BEA Tuxedo TOP END Domain Gateway.

e-docs Web Site

The BEA Tuxedo product documentation is available on the BEA Systems, Inc. corporate Web site. From the BEA Home page, click the Product Documentation button 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 BEA Tuxedo 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 BEA Tuxedo 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 installed, you can download it for free from the Adobe Web site at http://www.adobe.com/.

Related Information

For more information, see the BEA Tuxedo TOP END Domain Gateway online documentation.

Contact Us!

Your feedback on the BEA Tuxedo TOP END Domain Gateway 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 BEA Tuxedo documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA Tuxedo 8.0 release.

If you have any questions about this version of BEA Tuxedo, or if you have problems installing and running BEA Tuxedo, 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.

Convention	Item

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

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

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

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

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

Part I Basics

1 Introducing the BEA Tuxedo TOP END Domain Gateway

This topic includes the following sections:

- What Is the TOP END Domain Gateway?
- Understanding How the TOP END Domain Gateway Works

What Is the TOP END Domain Gateway?

BEA Tuxedo Domains provide a framework for interoperability between the domains of a business that require administrative autonomy but allow sharing of services and data. The *TOP END Domain Gateway* (TEDG) extends the BEA Tuxedo Domains concept to include BEA TOP END interoperability. The TEDG supports transactional message passing and queuing between the BEA TOP END and BEA Tuxedo systems.

Features of the TOP END Domain Gateway

The TOP END Domain Gateway provides the following features:

- Message Passing

- Queue Processing
- Transactional Support
- Security

Message Passing

Message passing provides bi-directional communication between BEA TOP END and BEA Tuxedo user components. The primary message passing capabilities are:

- BEA TOP END client applications can initiate messages that are transparently delivered to BEA Tuxedo servers.
- BEA Tuxedo client applications can initiate messages that are transparently delivered to BEA TOP END servers.
- Request/response mode is supported in both directions.
- BEA TOP END pseudo-conversational mode is mapped to BEA Tuxedo conversational mode.

Queue Processing

Queue processing capabilities include:

- BEA TOP END application components can enqueue items onto BEA Tuxedo /Q queues.
- BEA Tuxedo Application-to-Transaction Monitor Interface (ATMI) applications can enqueue items onto BEA TOP END RTQ queues.
- Processing of requests from the queues is handled by the system that hosts the queue.
  - Requests enqueued to BEA TOP END RTQ queues are processed by the RTQ subsystem. BEA TOP END RTQ does not support an explicit dequeue concept.
  - Requests enqueued to BEA Tuxedo /Q queues may be explicitly dequeued by an application or processed via `TMQFORWARD`.

Transactional Support

Transactional support allows global transactions to span BEA Tuxedo systems and BEA TOP END systems. That is, if a global transaction is begun in a component of one product and subsequent messages or enqueuing operations are sent to components in another product, the scope of the transaction includes work performed by components in the second product.

Security

The TOP END Domain Gateway provides the following security methods for communications:

- Authentication—each domain performs local authentication of clients.
- Authorization—existing BEA Tuxedo and Domains authorization is supported in the TOP END Domain Gateway.
- Encryption—the link between the TOP END Domain Gateway and the BEA TOP END system is optionally encrypted to ensure privacy.

See Also

- “Interconnecting Systems with the TOP END Domain Gateway” on page 2-1
- “What Is the BEA Tuxedo Domains Component?” on page 1-1 in *Using the BEA Tuxedo Domains Component*

Understanding How the TOP END Domain Gateway Works

This topic includes the following sections:

- BEA Tuxedo and BEA TOP END Terminology
- BEA Tuxedo Architecture Overview
- BEA TOP END Architecture Overview
- How the TEDG Works

BEA Tuxedo and BEA TOP END Terminology

While BEA Tuxedo and BEA TOP END offer similar capabilities, the terminology used by each product family is different. The following table provides a comparison of BEA TOP END and BEA Tuxedo terminology.

Table 1-1 Comparison of BEA Tuxedo and BEA TOP END Terms

BEA Tuxedo	BEA TOP END	Description
Domain	System	Collection of servers (programs) and other components that are deployed on a network of one or more nodes and administered together.
Bridge	Network Interface (NI)	Process that manages internode communication within a domain or system.
Bulletin Board Liaison (BBL)	Node Manager (NM)	A set of one or more processes that manage a domain or system on a run-time node.

Table 1-1 Comparison of BEA Tuxedo and BEA TOP END Terms (Continued)

BEA Tuxedo	BEA TOP END	Description
Workstation Handler (WSH)	Network Agent (NA)	Process that handles service requests from remote clients.
/Q	RTQ	A set of reliable queuing facilities that allow messages to be passed using queues.
ATMI	CSI	Application programming interfaces for the BEA Tuxedo and BEA TOP END systems, respectively.
Workstation client	Remote client	Client process running on a remote site.

TEDG Terminology

The following table summarizes common terminology used in reference to the TEDG.

This Term . . .	Refers to . . .
Remote domains	BEA TOP END systems accessed via the TEDG.
Remote domain `DOMAINID`	BEA Tuxedo user ID for requests made to the BEA Tuxedo system by the BEA TOP END system.
Local domains	BEA Tuxedo domains accessed via the TEDG. The term “local domains” can also be used to refer to a subset of the domain.
Local domain `DOMAINID`	BEA TOP END user ID for requests made to the BEA TOP END system by the BEA Tuxedo system.
Imported services	Remote services imported from a BEA TOP END system.
Exported services	Local services exported to a BEA TOP END system.

Reference Manual Notation

TEDG users should keep in mind that references to entries in the *BEA TOP END Programmer's Reference Manual* are marked by a "T" after the section number included in the reference. For example, the `tp_intro(1T)` entry can be found in the *BEA TOP END Programmer's Reference Manual*. Entries referenced without a "T" after the section number (such as `tmboot (1)`) can be found in the appropriate section of the BEA Tuxedo reference manual:

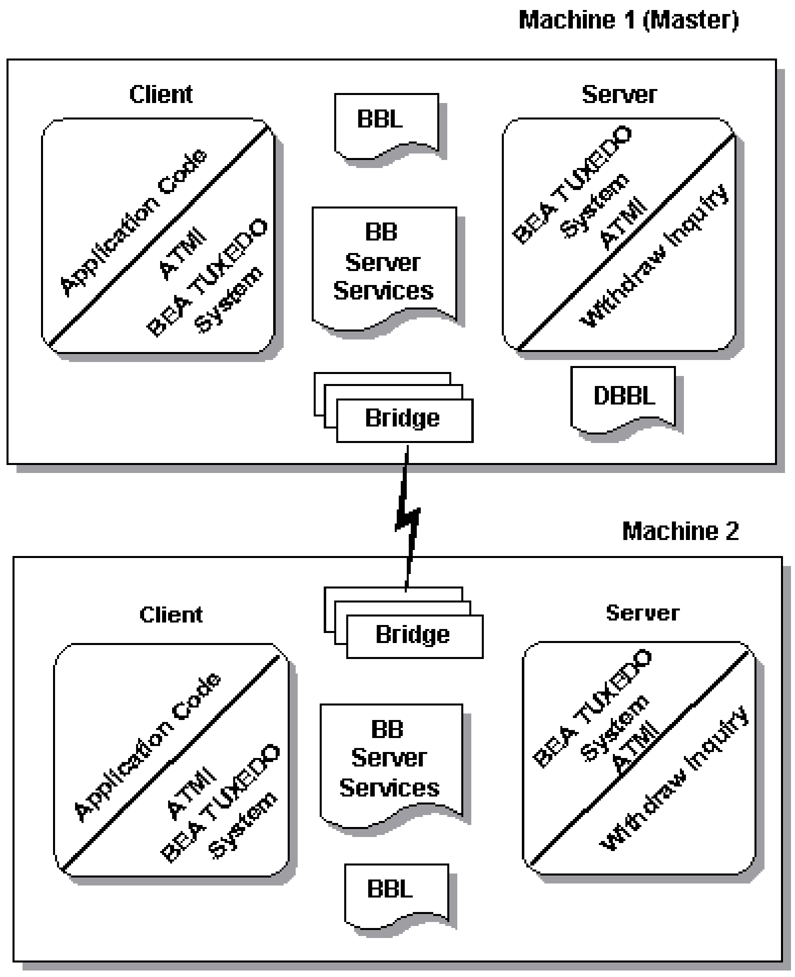
- *BEA Tuxedo Command Reference*
- *BEA Tuxedo ATMI C Function Reference*
- *BEA Tuxedo ATMI COBOL Function Reference*
- *BEA Tuxedo ATMI FML Function Reference*
- *File Formats, Data Descriptions, MIBs, and System Processes Reference*

BEA Tuxedo Architecture Overview

The BEA Tuxedo system uses the concept of a domain to define a collection of nodes (or machines) involved in an administratively autonomous application. A system administrator uses two files, `UBBCONFIG` and `DMCONFIG`, to configure a BEA Tuxedo domain.

A distributed-domain configuration consists of one or more business applications running on multiple machines. Although it contains multiple machines, this type of configuration is considered a single domain because it is administered centrally as a single entity. The following diagram shows the basic parts of a configuration distributed across multiple machines.

Figure 1-1 BEA Tuxedo Distributed Configuration



The primary components of a distributed configuration are:

- Client applications—executable programs that may request services through the BEA Tuxedo system.
- Server applications—executable programs that offer named services through the BEA Tuxedo system.

- ATMI—the application programming interface (API) for the BEA Tuxedo system.
- BEA Tuxedo Services—services and/or capabilities common to the BEA Tuxedo system infrastructure for developing and administering applications. The application processing services available to developers include: transactions, messaging paradigms, type validation, load balancing, data-dependent routing, service prioritization, data encoding, marshalling, compression, and reliable queuing. The administrative services include: distributed transaction processing, security management, service naming, distributed application administration, centralized application configuration, dynamic reconfiguration, and domains partitioning.
- Bulletin Board—a shared memory segment that holds configuration and dynamic information for the system. It is available to all BEA Tuxedo processes.
- Bulletin Board Liaison (BBL)—a BEA Tuxedo administrative process that monitors the data stored in the bulletin board and any changes made to it.
- Distinguished Bulletin Board Liaison (DBBL)—a process dedicated to making sure that the BBL server on each machine is alive and functioning correctly. This server runs on the master machine of a domain and communicates directly with all administration facilities.
- Bridges—BEA Tuxedo supplied servers that send and receive service requests between machines, and route them to local servers.

A multiple-domain configuration consists of two or more domains that communicate with each other through a gateway process in each domain. This gateway process provides bidirectional transaction control, and administrative tools that allow the configuration of the information required for interoperability of the local application with other domains. This configuration information includes the identification of a set of exported services, imported services, addressing, and access control. For an illustration of a multiple-domain configuration, see “What Is a Multiple-domain Configuration?” on page 3-47 in *Introducing BEA Tuxedo ATMI*.

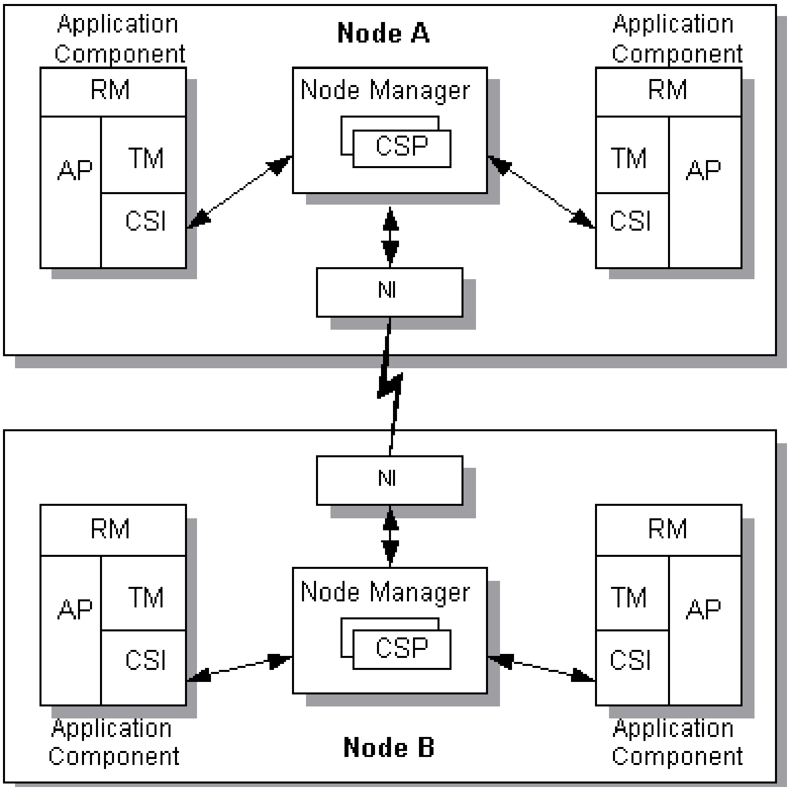
See Also

- “Interconnecting Systems with the TOP END Domain Gateway” on page 2-1
- “Basic Architecture of the BEA Tuxedo ATMI Environment” on page 2-1 in *Introducing BEA Tuxedo ATMI*

BEA TOP END Architecture Overview

The basic architecture of a BEA TOP END system is shown in the following figure.

Figure 1-2 Basic BEA TOP END System Architecture



As shown in the figure, a BEA TOP END system consists of one or more nodes. A basic node consists of a client, a node manager, and a server.

- A node may have one or more BEA TOP END systems.
- Each BEA TOP END system on a node has one node manager component.

An Application Component consists of the following:

- Application (AP)
- Resource Manager (RM)
- Transaction Manager (TM)
- Client/Server Interaction API (CSI)

The network interface (NI) is an optional component in a BEA TOP END system. An NI is not required for single-node systems. The Network Interface:

- Enables internode communication between components
- Provides optional internode message security through Kerberos routines

See Also

- "Interconnecting Systems with the TOP END Domain Gateway" on page 2-1
- *BEA TOP END System Design Guide*

How the TEDG Works

The TOP END Domain Gateway (TEDG) provides interoperability between BEA TOP END systems and BEA Tuxedo Domains. The TEDG consists of two parts: a gateway administrative server and a gateway process.

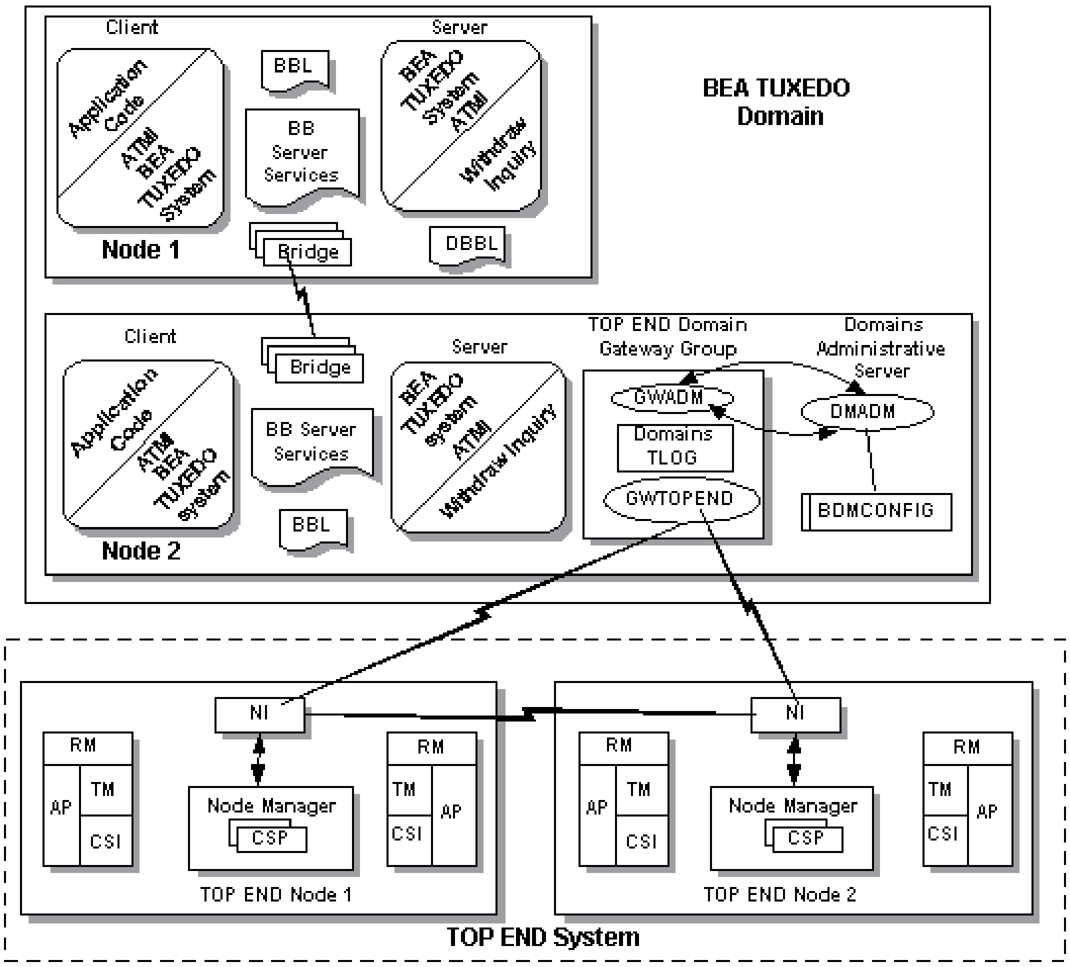
Part	Description
Gateway administrative server (`GWADM`)	The gateway group administrative server that registers with the `DMADM` server to obtain configuration information used by the gateway group.
Gateway process (`GWTOPEND`)	The gateway process that provides connectivity to remote BEA TOP END systems.

Each GWADM/GWTOPEND pair represents an instance of the TEDG. Multiple instances can be configured by specifying multiple GWADM/GWTOPEND pairs, one pair per BEA Tuxedo group.

- A TEDG connects to one or more nodes of a single BEA TOP END system. Connections must be made to each node that provides services required by a BEA Tuxedo application and to each node that uses BEA Tuxedo services. A TEDG can provide access to a BEA Tuxedo application only for connected BEA TOP END nodes.
- A TEDG can provide access to a BEA TOP END system for all BEA Tuxedo nodes in the domain or other connected domains.
- Multiple TEDG gateways may be configured in a domain or on a node to connect to different BEA TOP END systems.
- Multiple TEDG gateways may be configured to connect to a single BEA TOP END system as long as no more than one connection exists between any pair of nodes consisting of a TEDG node and a BEA TOP END node.
- The BEA TOP END system can communicate with multiple BEA Tuxedo domains as long as no more than one connection exists between any pair of nodes consisting of a TEDG node and a BEA TOP END node.
- Each TEDG is defined as a local domain in the DMCONFIG file that defines all gateways for a domain.

The following illustration shows how the TEDG connects a BEA TOP END system and a BEA Tuxedo domain.

Figure 1-3 Connecting Systems with the TOP END Domain Gateway



How a BEA Tuxedo User Views the TEDG

From the perspective of a BEA Tuxedo system user, the TEDG:

- Is controlled by the BEA Tuxedo system.

- Appears as a set of BEA Tuxedo services to the BEA Tuxedo client program.
- Allows BEA Tuxedo client programs to make calls to BEA TOP END services as if those services were part of the local domain.
- Maps requests from the BEA Tuxedo client programs to the appropriate BEA TOP END services, forwards those requests for processing, and then returns the results as required.
- Is viewed by a BEA Tuxedo client as a BEA Tuxedo server, and by a BEA Tuxedo server as a BEA Tuxedo client.

How a BEA TOP END User Views the TEDG

From the perspective of a BEA TOP END system user, the TEDG:

- Appears as another Network Interface (NI) in the BEA TOP END system.
- Allows BEA TOP END clients to access BEA Tuxedo services.
- Maps these requests from the BEA TOP END client programs to the appropriate BEA Tuxedo services, forwards them for processing, and then returns the results as required.
- Is viewed by a BEA TOP END client as a BEA TOP END server, and by a BEA TOP END server as a BEA TOP END client.

See Also

- "Interconnecting Systems with the TOP END Domain Gateway" on page 2-1

2 Interconnecting Systems with the TOP END Domain Gateway

This topic includes the following sections:

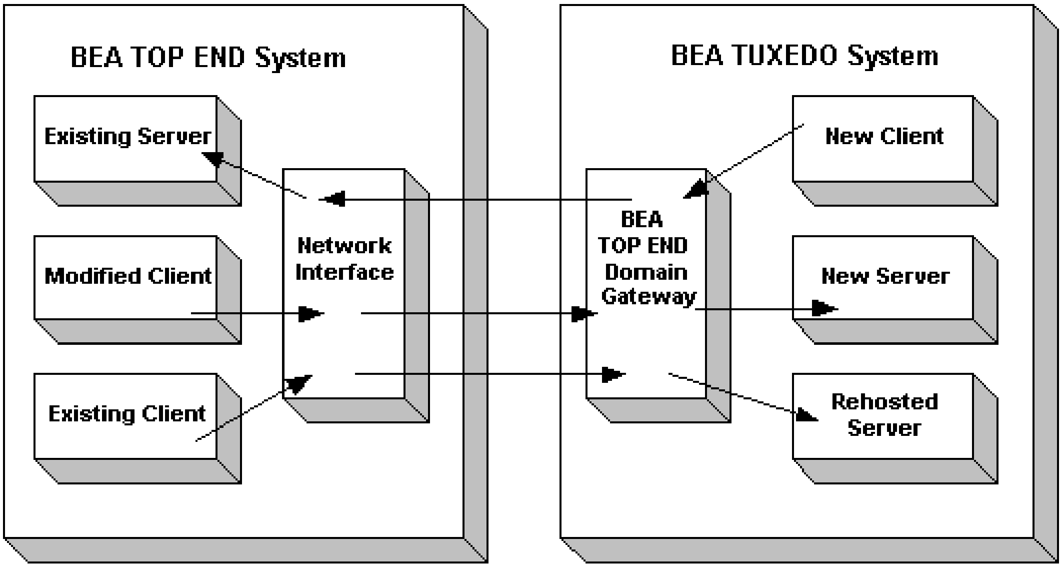
- Interoperability Scenarios Between the BEA TOP END and BEA Tuxedo Systems
- How the TEDG Handles Connections and Message Passing
- How the TEDG Handles Queue Processing
- How the TEDG Supports Transactions
- Characteristics of the APIs
- How the TEDG Is Administered

Interoperability Scenarios Between the BEA TOP END and BEA Tuxedo Systems

As shown in the following figure, the TOP END Domain Gateway supports three scenarios for interoperability between the BEA TOP END and BEA Tuxedo systems:

- A new BEA Tuxedo client communicating with existing BEA TOP END servers
- A modified BEA TOP END client communicating with a new BEA Tuxedo server
- An existing BEA TOP END client communicating with BEA TOP END services rehosted on an existing BEA Tuxedo server

Figure 2-1 Supported TEDG Interoperability Scenarios



See Also

- "Understanding How the TOP END Domain Gateway Works" on page 1-4

How the TEDG Handles Connections and Message Passing

This topic includes the following sections:

- Request/Response Message Passing
- Conversational Message Passing
- Establishing Connections Between BEA Tuxedo and BEA TOP END Systems
- Message Routing
- Message Size and Types

Request/Response Message Passing

The TEDG supports the request/response mode of passing messages between BEA Tuxedo and BEA TOP END systems. The systems use the TEDG as an intermediary.

BEA Tuxedo Client to BEA TOP END Server

BEA Tuxedo clients view the TEDG as a local server. The TEDG advertises services based on the `SERVICE` entries in the `DM_REMOTE_SERVICES` section of the `DMCONFIG` file. Client programs make requests using `tpcall(3c)` as if they were addressing a local service. Asynchronous requests and responses are supported via the `tpacall(3c)` and `tpgetrply(3c)` functions, respectively.

The TEDG uses the service name to locate the `SERVICE` entry in the `DM_REMOTE_SERVICES` section to determine the corresponding BEA TOP END product, function, MSR target, and function qualifier. For FML32 buffers, data marshalling is performed, as required, before a message is sent.

BEA TOP END servers respond to client requests using `tp_server_send(3T)`.

BEA TOP END Client to BEA Tuxedo Server

BEA TOP END clients request BEA Tuxedo services using `tp_client_send(3T)` or `tp_client_signon(3T)`. BEA TOP END client calls are asynchronous. Responses are handled by the `tp_client_receive(3T)` call.

Message routing is based on three pieces of information advertised by the TEDG: the product, the function and, optionally, the MSR target. Normal BEA TOP END routing and load-balancing rules are used to determine the destination node. When a message is received by the TEDG, the product, function, optional MSR target, and function qualifier specified in the message are used to determine the corresponding BEA Tuxedo service as specified by the `SERVICE` entries in the `DM_LOCAL_SERVICES` section of the `DMCONFIG` file. The message is sent to the appropriate BEA Tuxedo service after any required data unmarshalling is performed.

See Also

- `tpacall(3c)` in *BEA Tuxedo ATMI C Function Reference*
- `tpcall(3c)` in *BEA Tuxedo ATMI C Function Reference*
- `tpgetrply(3c)` in *BEA Tuxedo ATMI C Function Reference*
- `tp_client_receive(3T)` in *BEA TOP END Programmer's Reference Manual*
- `tp_client_send(3T)` in *BEA TOP END Programmer's Reference Manual*
- `tp_client_signon(3T)` in *BEA TOP END Programmer's Reference Manual*
- `tp_server_send(3T)` in *BEA TOP END Programmer's Reference Manual*

Conversational Message Passing

The BEA Tuxedo and BEA TOP END systems support different styles of conversational message passing.

- The BEA TOP END system supports the notion of pseudo-conversations. A BEA TOP END server maintains context (a conversation) with a client when the `TP_APPL_CONTEXT` flag is set. (See `tp_server_send(3T)` in the *BEA TOP END Programmer's Reference Manual* for details.)
- BEA Tuxedo conversations are more structured. To be conversational, a server must be configured as such by an administrator. Then a dedicated function (`tpconnect(3c)`) must be used to establish a connection and maintain the conversation. BEA Tuxedo conversations are half-duplex: one side of a conversation can send multiple messages before relinquishing control of the connection. (See "Introduction to the C Language Application-to-Transaction Monitor Interface" in the *BEA Tuxedo ATMI C Function Reference* for details.)

The TEDG provides a mapping of BEA TOP END pseudo-conversations to the BEA Tuxedo conversation model. This mapping allows:

- Existing BEA TOP END servers to maintain context with new BEA Tuxedo clients
- Existing BEA TOP END clients to engage in multi-step interactions with new BEA Tuxedo servers

Establishing Connections Between BEA Tuxedo and BEA TOP END Systems

BEA TOP END nodes advertise available services to adjacent nodes at connection time. New services are advertised dynamically. Services are unadvertised if they are no longer available. Thus, at any time a BEA TOP END node can determine the status of all services on nodes to which it is connected.

The BEA Tuxedo system takes a different approach. Within a multi-node BEA Tuxedo domain, the connection architecture is quite similar to that of a BEA TOP END system. However, interdomain connectivity is different.

The BEA Tuxedo system offers three possible connection policies that can be assigned to a TEDG:

- ON_STARTUP
- ON_DEMAND
- INCOMING_ONLY

By default, the connection to another domain is not made until a service request that requires such a connection is made (`CONNECTION_POLICY=ON_DEMAND`). This type of connection is called a lazy connection. The gateway, however, advertises remote services to the local domain before any connections are made. If a connection cannot be made, the gateway returns an error to the caller, but the relevant services remain advertised.

Optionally, a TEDG can be configured to connect, automatically, with other BEA TOP END systems at the start of each day (`CONNECTION_POLICY=ON_STARTUP`). If a connection is established, all associated services are assumed to be available and are advertised. If a connection cannot be established, or if a connection is established but subsequently fails, related services are suspended (`CONNECTION_POLICY=INCOMING_ONLY` and `CONNECTION_POLICY=ON_STARTUP`). Thus, remote services can be configured such that they are continually advertised (`CONNECTION_POLICY=ON_DEMAND`), or advertised only when a connection to the remote domain is open (`CONNECTION_POLICY=ON_STARTUP` and `CONNECTION_POLICY=INCOMING_ONLY`).

For local services, regardless of the connection policy, the TEDG advertises (exports) its local services to the BEA TOP END system when a connection is established with a BEA TOP END node. The TEDG determines which services to advertise using the rules in the `DMCONFIG` file for the local domain. The TEDG advertises each `SERVICE` entry in the `DM_LOCAL_SERVICES` section that is associated with the local domain. For each of these services the TEDG advertises the associated parameters (`TE_PRODUCT`, `TE_FUNCTION`, and `TE_TARGET`) to the BEA TOP END system. The status of individual service availability is not tracked by the BEA TOP END system; only the status of the network connection is tracked.

Connecting at Boot Time (ON_STARTUP)

The BEA Tuxedo `ON_STARTUP` policy most closely resembles the BEA TOP END connection policy. When this policy has been selected, the TEDG attempts to connect to each of its configured BEA TOP END nodes. If multiple network addresses are configured they are tried serially. Connection retries apply if configured. If a connection attempt succeeds, the TEDG advertises the associated remote services to the local domain. If a connection attempt fails (or is lost), the associated imported services are suspended until the connection is reestablished. The status of individual service availability on the BEA TOP END node is not tracked.

Connecting When a Client Program Requests a Remote Service (ON_DEMAND)

The `ON_DEMAND` or “lazy connection” policy is the default connection policy on the BEA Tuxedo system. Connections are attempted as needed to fulfill client requests for remote services. If multiple network addresses are configured they are tried serially. Connection retries do not apply. The TEDG advertises remote services to the BEA Tuxedo system on startup but does not actually connect to any BEA TOP END nodes. Remote services are always advertised, regardless of the connection status. The status of individual service availability on the BEA TOP END node is not tracked.

Accepting Incoming Connections Only (INCOMING_ONLY)

The `INCOMING_ONLY` policy specifies that the TEDG does not attempt to connect to remote domains and does not advertise associated remote services. Connections must be initiated by the BEA TOP END system connecting to the TEDG or by a manual connection attempt by the BEA Tuxedo administrator. Multiple network addresses can be handled only by an administrator establishing a connection manually. Connection retries do not apply. When a connection is successfully established, the associated remote services are advertised. If a connection fails, the associated imported services are suspended until the connection is reestablished. The status of individual service availability on the BEA TOP END node is not tracked.

Message Routing

Both the BEA TOP END and BEA Tuxedo systems support routing based on message content. However, the two facilities differ in the granularity of routing provided through the TEDG. You can use the data-dependent routing (DDR) feature of the BEA Tuxedo system and the message sensitive routing (MSR) feature of the BEA TOP END system to direct message traffic, at a low level, between both systems.These facilities are not mutually exclusive and can be used together to control message traffic between systems.

BEA Tuxedo Data-Dependent Routing (DDR)

Specify routing criteria for the TEDG, as you do for other Domains components, in the `DM_ROUTING` section of the `DMCONFIG` file. The criteria you specify allow messages to be routed to specific remote domains depending upon the content of the messages. For the TEDG, a remote domain maps to a specific NI (BEA TOP END system node).

BEA TOP END Message Sensitive Routing (MSR)

BEA TOP END clients can use MSR facilities or explicitly specify an MSR target as part of their service request. Because BEA Tuxedo services are mapped to BEA TOP END service parameters that include the optional MSR target (`TARGET` parameter in the `DM_LOCAL_SERVICES` section of the `DMCONFIG` file), it is possible to route messages from the BEA TOP END system to specific services provided by the BEA Tuxedo system (as long as the BEA Tuxedo services are mapped uniquely).

Formats and Message Content Conversion (MCC)

The TOP END Domain Gateway supports BEA TOP END clients and servers using either raw data buffers or FML32. The BEA TOP END formats facility, which is used primarily to describe screen formats, is not supported.

The Message Content Conversion (MCC) feature, which provides hooks for message translation and relies on formats, is also not supported. When a message is sent to the TEDG with a format specified, the format information is ignored and a message about this occurrence is recorded in the `userlog`.

Message Size and Types

The BEA TOP END system and the TEDG can send and receive messages up to 30K (30 X 1024) bytes (30,000 for RTQ messages) long. BEA TOP END Large Message Architecture (LMA) is not supported by the TEDG.

You can disable the LMA feature in the BEA TOP END application programming interface by setting the `attach_info` parameter to `NULL` in all BEA TOP END CSI calls. If an attachment is included in a function call, the TEDG will:

- Document the condition of the call in the `userlog`
- Discard the attachment
- Process the request or response message

Because the BEA Tuxedo system allows large messages as part of normal processing, the TEDG detects large messages, returns an error to the client, and does not forward the messages to the BEA TOP END system.

The TEDG supports BEA TOP END clients and servers using either raw data buffers or FML32. The use of BEA TOP END screen formats is not supported.

How the TEDG Handles Queue Processing

Both the BEA TOP END and BEA Tuxedo systems support reliable queues as a means of passing messages between components. The BEA Tuxedo /Q facility and the BEA TOP END Recoverable Transaction Queuing (RTQ) product allow messages to be passed using a queue as a stable intermediary, capable of storing messages before processing. This model is useful when offline processing of requests is preferred, or when such processing is the only practical method of client/server communication. Both /Q and RTQ guarantee that once a message is successfully placed on a queue it will be delivered to a server. Full transaction semantics are supported for both the original message queuing and the subsequent message processing.

The TEDG supports the queuing of messages between the BEA Tuxedo and BEA TOP END systems, as well as transactional queuing between systems. While /Q and RTQ offer similar capabilities, they are fundamentally different in their approach to handling queued messages.

The BEA Tuxedo system provides a function that allows programs to explicitly dequeue messages (see `tpdequeue(3c)` in *BEA Tuxedo ATMI C Function Reference*). Alternatively, a system-supplied service, `TMQFORWARD(5)`, can be configured to dequeue messages automatically and forward them to standard BEA Tuxedo servers via the normal `tpcall(3c)` function. The destination service is mapped to the queue name.

In contrast, the BEA TOP END system does not provide a dequeuing facility for user programs. Instead, messages are dequeued by the RTQ subsystem, which delivers them to the intended service. The destination service address is supplied as part of the process of enqueuing the message (see `tp_rtq_put(3T)` in *BEA TOP END Programmer's Reference Manual*). The relationship between the queues and the services is arbitrary.

See Also

- `tpcall(3c)` in *BEA Tuxedo ATMI C Function Reference*
- `TMQFORWARD(5)` in *File Formats, Data Descriptions, MIBs, and System Processes Reference*
- "Reliable Queuing Programming" on page 14-1

How the TEDG Supports Transactions

The TOP END Domain Gateway (TEDG) supports transactional message passing and queuing between BEA TOP END and BEA Tuxedo systems.

Specifically, the TEDG performs the following functions:

- Advertises the *Transaction Manager Server* (TMS) service and performs the associated transaction management functions currently performed by other Domains products.
- Can act as a subordinate of transactions initiated by another client or server in the local domain.
- Appears as the superior for transaction branches on remote nodes.
- Can act as a subordinate for transactions coordinated by the BEA TOP END system. In this capacity the TEDG also serves as the coordinator for all server groups in the local domain.

Transaction Identifier Mapping

Both BEA TOP END and BEA Tuxedo systems follow the XA standard for transaction identifiers (`XID`). In the BEA Tuxedo system, the latter are referred to as Global Transaction Identifiers (`GTRID`). BEA TOP END users identify transactions with the term "XID." The XA XID is composed of a Global Transaction ID (`GTRID`) and a Branch Qualifier (`BQUAL`). The TEDG creates a new transaction branch for each new transaction step; this is known as a loosely-coupled relationship. Refer to "Using GTRID Mapping in Transactions" on page 4-13 in *Using the BEA Tuxedo Domains Component* for details on loosely-coupled and tightly-coupled relationships. The TEDG supports loop-back transactions that span BEA TOP END or BEA Tuxedo systems multiple times. A new branch is generated for each hop.

For transaction branches destined for the BEA TOP END system, the TEDG creates a branch following the BEA TOP END branch design. For incoming transaction branches the BEA Tuxedo design is used.

Transaction Management

The TEDG supports transaction management by providing transaction logging and recovery capabilities.

Logging Transactions

The TEDG logs the transaction state during the two-phase commit process to enable recovery. Each instance of the TEDG has its own log. The specification of the `DMTLOG` is given in the `DM_LOCAL_DOMAINS` section of the `DMCONFIG` file. As with other Domains features, the logging functionality is provided by the `GWADM` administrative server.

When a transaction has been committed on all nodes, the records for that transaction are removed from the log.

Transaction Recovery

The TEDG performs recovery in a fashion identical to other Domains products.

When a domain gateway group is booted, the gateway performs an automatic warm start of the `DMTLOG`. The warm start includes a scan of the log to determine whether any transactions have not been completed. If incomplete transactions are found, action is taken to complete them.

- For a prepared transaction: The TEDG contacts the superior node for the transaction to determine whether the transaction should be committed or rolled back.
- For a committed transaction: The TEDG ensures that all subordinates have committed the transaction.
- Any transactions that are still in progress are rolled back.

The TEDG supports connections to BEA TOP END nodes that use the BEA TOP END Extended Recovery (XR) feature. The TEDG itself cannot be an XR node.

See Also

- "How the TEDG Handles Connections and Message Passing" on page 2-3
- "Transactional Support Programming" on page 15-1

Characteristics of the APIs

Both the BEA Tuxedo and BEA TOP END provide application programming interfaces (APIs) that follow the X/Open DTP model. The BEA Tuxedo API is called the Application-to-Transaction Monitor Interface (ATMI). The BEA TOP END API is referred to as the Client/Server Interaction (CSI) facility.

Characteristics of the BEA Tuxedo ATMI

The BEA Tuxedo ATMI provides the interface between the application and the transaction processing system. It provides routines to open and close resources, manage transactions, manage typed buffers, and invoke request/response, conversational, and queuing functions.

The BEA Tuxedo ATMI functions are used by BEA Tuxedo client and server programs to communicate with BEA TOP END client and server programs through the TOP END Domain Gateway (TEDG). To a great extent, the way in which ATMI functions are used to communicate with BEA TOP END applications is identical to how ATMI functions are used to communicate with other BEA Tuxedo programs or applications through other domain gateways. The primary difference is that the TEDG does not support the same functions and features as other domain gateways. Because the TEDG connects two similar but not identical client/server environments, only features common to both environments can be supported when these environments are interoperating. This set of features common to both the BEA Tuxedo and BEA TOP END systems is provided by the TEDG through configuration, feature mapping, and exposing supported features at the ATMI programming interface in a way that closely matches the standard use of the ATMI.

Supported ATMI Features and Functions

The following ATMI programming features are supported for communication via the TEDG. The functions listed as examples are the primary functions associated with the features from a TEDG perspective. To write a complete application, other interface functions are also required.

- Request/response communication (for example, `tpcall(3c)`, `tpacall(3c)`, `tpgetrply(3c)`, `tpreturn(3c)`, and `tpforward(3c)`)
- Conversational communication (for example, `tpconnect(3c)`, `tpsend(3c)`, `tprecv(3c)`, `tpdiscon(3c)`, and `tpreturn(3c)`)
- Queuing (for example, `tpenqueue(3c)`)
- Transaction management (for example, `tpbegin(3c)`, `tpcommit(3c)`, `tpabort(3c)`, and `tx_*` equivalents)
- Buffer types `FML32`, `CARRAY`, and `X_OCTET`
- Standard return values and `tperrno` values (although the meanings of some values may vary, depending on gateway conditions)
- Transactions that involve both a BEA Tuxedo application and a BEA TOP END system
- Security
- Programming and BEA Tuxedo system application features, such as priorities and timeouts, that affect only the behavior of the BEA Tuxedo system and therefore do not require direct support in the remote BEA TOP END system

Unsupported ATMI Features

For several BEA Tuxedo ATMI features there are no corresponding features in the BEA TOP END system. Therefore, these features are not supported by the TEDG, although they are still available in the BEA Tuxedo system. The following ATMI features are not supported by the TEDG:

- Unsolicited notification
- EventBroker
- Dynamic service advertisement

- /Q reply and failure queues
- Conversations with multiple consecutive messages in a single direction
- Administration of BEA TOP END systems through BEA Tuxedo administration interfaces

Characteristics of the BEA TOP END CSI

CSI, the BEA TOP END application programming interface (API), provides a set of routines used by BEA TOP END client and server programs to communicate with BEA Tuxedo client and server programs through the TEDG. To a great extent, the way in which CSI routine calls are used to communicate with BEA Tuxedo ATMI applications is identical to how CSI routines are used to communicate with other CSI applications. The primary difference is the features or options supported for each routine. The TEDG supports a set of features common to both the BEA Tuxedo and BEA TOP END systems through configuration, feature mapping, and exposure of supported features at the CSI programming interface in a way that closely matches the standard use of the CSI.

Supported CSI Features and Routines

BEA TOP END offers several programming interfaces. For the purpose of this document, they can all be considered part of CSI (Client/Server Interaction). The following programming features and routines are supported for communication via the TEDG:

- Request/response and conversational (for example, `tp_client_signon`, `tp_client_send`, `tp_client_receive`, `tp_server_receive`, and `tp_server_send`)
- Recoverable Transaction Queuing (RTQ) (for example, `tp_rtq_put`)
- Transactional (for example, `tx_open`, `tx_begin`, `tx_commit`, `tx_rollback`, `tx_close`, `tp_begin`, `tp_commit`, and `tp_rollback`)
- Raw buffers and FML32 buffers (FML32 is supported in BEA TOP END 3.0 systems only)
- Standard status values and extended status values (although the meanings of some values may vary, depending on gateway conditions)

- Transactions that involve both a BEA Tuxedo application and the BEA TOP END system
- Security
- Features unique to the BEA TOP END system that do not require direct support in the BEA Tuxedo system, such as Message Sensitive Routing (MSR)

The routine calls listed as examples are the primary routine calls associated with the features from a TEDG perspective. To write a complete application, other routine calls are also required. In addition, other variations of the calls listed here may be available.

Unsupported CSI Features

For several BEA TOP END programming facilities there are no corresponding features in the BEA Tuxedo system. Therefore these programming facilities are not supported by the TEDG, although they are still available in the BEA TOP END system.

- Formats and Message Content Conversion (MCC)
- Large Message Architecture (LMA)
- Administration of the BEA Tuxedo system through BEA TOP END administration interfaces (SMAPI)

See Also

- "API Programming" on page 11-1

How the TEDG Is Administered

The TOP END Domain Gateway (TEDG) is administered as part of the BEA Tuxedo application space using standard BEA Tuxedo tools (`tmadmin(1)` and `dmadmin(1)`). The TEDG cannot be administered through the BEA TOP END system.

The TEDG is primarily configured using BEA Tuxedo tools (`DMCONFIG` and `dmloadcf(1)`). The only configuration necessary in the BEA TOP END system, beyond configuring the TEDG nodes, is configuration of any security mechanisms being used.

Run-time administration of the TEDG is done using BEA Tuxedo tools: the `dmadmin` command, and the `DMADM` and `GWADM` gateway servers.

See Also

- "Configuring the TOP END Domain Gateway" on page 4-1
- "Configuring Security Between BEA TOP END and BEA Tuxedo Systems" on page 7-1
- "Administering the TEDG During Run Time" on page 10-1

CHAPTER

3 Understanding Security Between the BEA TOP END and BEA Tuxedo Systems

This topic includes the following sections:

- Overview of BEA TOP END Security
- Overview of BEA Tuxedo Security
- How Security Is Handled by the TEDG
- How the TEDG Establishes a Secure Connection to the NI
- How BEA Tuxedo to BEA TOP END Security Works
- How BEA TOP END to BEA Tuxedo Security Works

See Also

- "Configuring Security Between BEA TOP END and BEA Tuxedo Systems" on page 7-1

Overview of BEA TOP END Security

Enabling BEA TOP END security means that the BEA TOP END system performs user authentication, user authorization, and node authentication at startup and whenever messages are sent or received. These features cannot be enabled individually. The security realm for a BEA TOP END application is the BEA TOP END system. All nodes in a system must have identical security configurations. When two nodes attempt to connect, the security configuration is checked on both nodes. If security is not configured identically on the two nodes, the connection is refused. When security is enabled, BEA TOP END Node Managers (NM) authenticate each other as part of the connection process.

For the TEDG, BEA TOP END security is enabled by the `SECURITY` parameter in the `DM_LOCAL_DOMAINS` section of the `DMCONFIG` file, and the `nm_config` file on the BEA TOP END node.

Authentication and Authorization

If BEA TOP END security is enabled, then all clients are authenticated by `tp_client_signon(3T)` and all subsequent requests for service are checked for authorization. Authentication and authorization work together; they cannot be separated. Authorization is performed on a product and function basis.

Message Protection/Encryption

If BEA TOP END security is enabled, then messages between NIs may be sent in one of the following ways:

- clear text (`CLEAR`)
- protected by checksum (`SAFE`)
- encrypted (`PRIVATE`)

Kerberos 4 is used to protect internode messages. The same message protection level is required for all connections within the BEA TOP END system. However, a separate key is created for each connection as part of the connection process. This feature is supplied by the BEA TOP END system; it cannot be replaced by the customer.

Overview of BEA Tuxedo Security

A BEA Tuxedo domain may be configured with several levels of security. For details about the various levels of security available for BEA Tuxedo Application-to-Transaction Monitor Interface (ATMI) applications, see `UBBCONFIG(5)` in the *File Formats, Data Descriptions, MIBs, and System Processes Reference.*

Authentication/Authorization

You can authenticate a client in either of two ways. You can:

- Define an application-wide password (`APP_PW`)
- Specify individual client authentication (`USER_AUTH`)

The BEA Tuxedo system provides proprietary authentication and authorization services. Authentication is based on a user ID and password for each user. Authorization is based on Access Control Lists (ACLs), which specify the users entitled to access particular resources (services, queues, and events).When a user requests use of a resource, the system searches for an ACL for that resource. If an ACL is found, the system checks it to determine whether the user is authorized to use the resource. The strongest level of security requires explicit authorization (`MANDATORY_ACL`) for access to any service, queue, or event.

Optional Encryption

Optional encryption can be configured to protect data between nodes. Unlike BEA TOP END encryption, BEA Tuxedo encryption can be enabled without user authentication and authorization.

Public Key Encryption

There are two types of public key encryption used in BEA Tuxedo ATMI applications: message-based encryption and message-based digital signature. Both build on the technology and key management of public/private key encryption algorithms.

Both message-based encryption and message-based digital signatures for application messages are supported between the BEA Tuxedo application and the TEDG but do not apply to messages between the TEDG and BEA TOP END systems.

System Interoperability

The BEA Tuxedo system allows domains to interoperate through domain gateways. Because domains are configured independently, any two domains do not need to have the same security configurations. Gateways provide configuration options that allow administrators to control the level of interoperability between any two domains.

Interdomain Security

Four levels of security are provided by a domain gateway, as specified in the `DMCONFIG` file:

- Gateways can be configured to limit the set of local services made available to remote domains
- Access to specific services can be limited to select remote domains
- Gateways can be configured to require authentication when attempts are made to connect to remote domains
- Links between domains may be encrypted

See Also

- "Introducing ATMI Security" on page 1-1 in *Using Security in CORBA Applications*

How Security Is Handled by the TEDG

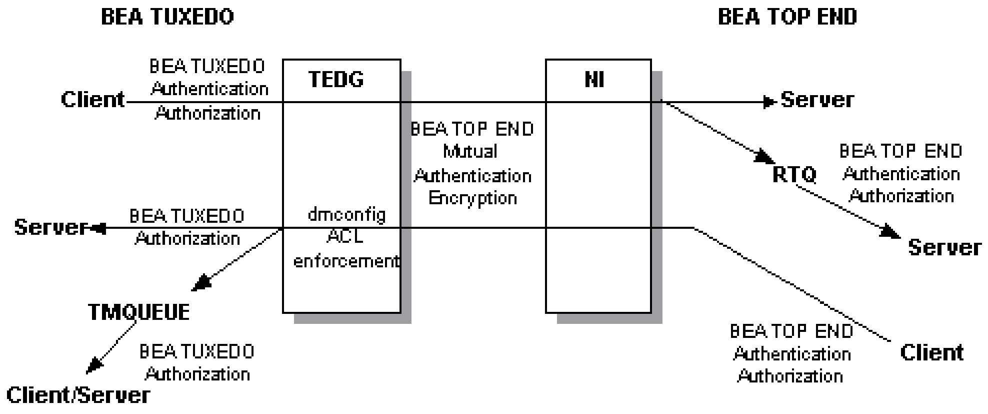
The TEDG handles security in a manner similar to how the BEA Tuxedo TDomain handles security.

- The local BEA Tuxedo domain or BEA TOP END system controls access for local clients. Clients log in once to their respective systems, and are authenticated and authorized according to the local policy.
- The TEDG allows an administrator to control access to BEA TOP END services by BEA Tuxedo clients. Similarly, the administrator can control access to BEA Tuxedo services from the BEA TOP END system.
- Requests passing through the gateway are performed using the credentials of the gateway. This practice eliminates the need to maintain duplicate security databases on both the BEA Tuxedo and BEA TOP END systems.
- Mutual authentication of the BEA TOP END and TEDG nodes and encryption of messages between nodes is optional. These capabilities, based on the BEA TOP END security algorithms, use the BEA TOP END security package.

The following illustration shows how security works between BEA Tuxedo and BEA TOP END systems. For more detailed information, see:

- "How BEA Tuxedo to BEA TOP END Security Works" on page 3-6
- "How BEA TOP END to BEA Tuxedo Security Works" on page 3-8

Figure 3-1 TEDG to BEA TOP END Security


BEA TUXEDO
BEA TOP END
BEA TUXEDO Authentication Authorization
Client
TEDG
NI
Server
BEA TOP END Mutual Authentication Encryption
RTQ
BEA TOP END Authentication Authorization
Server
BEA TUXEDO Authorization
dmconfig ACL enforcement
TMQUEUE
BEA TUXEDO Authorization
Client/Server
BEA TOP END Authentication Authorization
Client

How BEA Tuxedo to BEA TOP END Security Works

Clients are authenticated and authorized by the BEA Tuxedo system on the basis of how the local domain is configured in the `UBBCONFIG(5)` file. If BEA TOP END security is enabled, an additional security check can be done on the BEA TOP END node.

BEA Tuxedo-side Security

Clients are authenticated by the BEA Tuxedo system in the same way as any other BEA Tuxedo client, via user ID and password. Clients are authorized through a standard BEA Tuxedo authorization scheme characterized by the following:

- The TEDG advertises the `SERVICE` and `QSPACE` entries listed in the `DM_REMOTE_SERVICES` section of the `DMCONFIG` file as services to the local BEA

Tuxedo domain. BEA TOP END services not placed in the `DMCONFIG` file are not accessible by BEA Tuxedo clients.

- Access to services advertised by the TEDG is controlled by the local BEA Tuxedo domain configuration. The level of authorization control is determined by the value of the `SECURITY` parameter in the `UBBCONFIG` file for the domain. A value of `ACL` or `MANDATORY_ACL` indicates that the BEA Tuxedo system should perform access control checks. A BEA Tuxedo administrator can create ACLs for imported BEA TOP END services.

If the BEA Tuxedo-side security is successful, the TEDG prepares to send the message to the BEA TOP END system. At this point, BEA TOP END security takes over.

BEA TOP END-side Security

If BEA TOP END security is enabled, the TEDG inserts a user ID into all messages passed to the BEA TOP END system. To enqueue requests, the TEDG provides both a user ID and password in each message. The password is protected using the current BEA TOP END algorithm used by RTQ.

The TEDG uses a single set of credentials for all messages passed to the BEA TOP END system:

- The TEDG TOP END user ID (the `DOMAINID` in the `DM_LOCAL_DOMAINS` section of the `DMCONFIG` file)
- A password established through the `topendpasswd` command in the `dmadmin(1)` utility

The password is stored in the `BDMCONFIG` file in an encrypted format. The administrator defines a matching user ID and password in the BEA TOP END security database using the BEA TOP END `tpsecure(1T)` utility.

If BEA TOP END security is enabled, BEA TOP END message passing requires that messages carry the user ID of the client. Because the user ID is not reauthenticated by the BEA TOP END system, a password is not required; the user ID is provided purely for information. For queuing, the BEA TOP END system requires that both the user ID and password be passed along with the service request. The BEA TOP END system uses these credentials to authenticate the client while processing the queued service request.

The BEA TOP END system does not perform any additional access control checking for message passing requests. However, queued requests are authorized by the BEA TOP END system when they are retrieved by RTQ and the service request is processed. Because all messages from the TEDG are submitted using the TEDG TOP END user ID (that is equal to the local domain `DOMAINID`), this TEDG user ID must be authorized to perform the requested service. The administrator must create ACLs for the TEDG user ID using the TOP END `tpsecure(1T)` utility.

See Also

- "Configuring Security Between BEA TOP END and BEA Tuxedo Systems" on page 7-1

How BEA TOP END to BEA Tuxedo Security Works

Clients are authenticated and authorized by BEA TOP END, based on the configuration of the BEA TOP END system. If BEA Tuxedo security is enabled, an additional security check can be done on the BEA Tuxedo node.

BEA TOP END-side Security

If BEA TOP END security is enabled, clients are authenticated at signon. The TEDG does not perform client authentication on incoming requests.

The BEA TOP END system performs authorization checks on the client node before a message is sent. If BEA TOP END security is enabled, the client must be granted authorization to access the requested BEA Tuxedo service or queue. The administrator must create ACLs using the BEA TOP END `tpsecure(1T)` utility for each BEA TOP END user who accesses BEA Tuxedo resources. The BEA TOP END products and functions must match the entries in the `DM_LOCAL_SERVICES` section of the `DMCONFIG` file used to map BEA TOP END resources to the BEA Tuxedo system.

BEA Tuxedo-side Security

The TEDG provides the following levels of access control for incoming requests from the BEA TOP END system:

- In the first level of access control, the `DM_LOCAL_SERVICES` section of the `DMCONFIG` file lists the BEA Tuxedo `SERVICE` entries and `QSPACE` entries that are to be advertised to the BEA TOP END system.
- In the second level of access control, the domain gateway access control feature can be used to limit access to particular services from specific BEA TOP END nodes (or NI connections).

  The administrator can specify an ACL parameter as part of the `DM_LOCAL_SERVICES` entry for a specific `SERVICE` or `QSPACE` entry. The ACL is specified in the `DM_ACCESS_CONTROL` section of the `DMCONFIG` file. Each ACL record contains the names of remote domains allowed to access the service. The remote domains are mapped to BEA TOP END NI instances in the `DM_REMOTE_DOMAINS` section of the configuration file. Using ACL entries is useful for limiting access to advertised services to specific BEA TOP END nodes.
- In the third level of access control, normal BEA Tuxedo authorization may be performed.

  If a request passes TEDG authorization, normal BEA Tuxedo authorization is performed. If `ACL` or `MANDATORY_ACL` is specified in the `UBBCONFIG(5)` file, then the `DOMAINID` of the remote domain making the request is used as the BEA Tuxedo username. If `SECURITY=ACL` in the `UBBCONFIG` and there is an entry in the `ACL` database for this service, the entry must include the `DOMAINID` of the `RDOM`; otherwise, the service or enqueue request fails. If `SECURITY=MANDATORY_ACL` in the `UBBCONFIG`, there must be an entry in the ACL database for this service, and the entry must include the `DOMAINID` of the `RDOM`; otherwise, the service or enqueue request fails.

See Also

- "Understanding Security Between the BEA TOP END and BEA Tuxedo Systems" on page 3-1

How the TEDG Establishes a Secure Connection to the NI

All nodes in a BEA TOP END system must be configured for the same level of message protection. The `SECURITY` parameter in the `DM_LOCAL_DOMAINS` section of the `DMCONFIG` file determines the level of protection configured for the TEDG. Three levels of protection are available: `CLEAR`, `SAFE`, and `PRIVATE`.

- At the lowest level, `CLEAR`, messages are sent without any protection.
- At the next level, `SAFE`, a `checksum` for each message is generated by the Kerberos 4 libraries and is appended to the message.
- At the highest level, `PRIVATE`, the Kerberos libraries, using the DES algorithm, encrypt each message. An encryption key is automatically established when the TEDG and NI connect.

These `SECURITY` parameter values correspond to the BEA TOP END Node Manager (NM) configuration parameters `[security]` and `[internode security]` as described in `nm_config(4T)` in *BEA TOP END Programmer's Reference Manual*.

When started, the TEDG checks the configuration to determine whether security is enabled (that is, to determine whether a value of `CLEAR`, `SAFE`, or `PRIVATE` has been assigned to the `SECURITY` parameter). If it is enabled, the TEDG needs a Kerberos Ticket Granting Ticket (TGT), just as the BEA TOP END NM does at start of day. To obtain a TGT, the Kerberos database used by the BEA TOP END system must contain an entry (Principal) for the machine on which the TEDG is running. If the TEDG cannot obtain a TGT, the TEDG logs an error and terminates.

The TEDG to NI connection process follows the BEA TOP END sign-on protocol. As part of this protocol, the TEDG and NI exchange security configurations and check to make sure the two configurations match exactly. If the configurations do not match, the TEDG logs an error (see `userlog(3c)` in *BEA Tuxedo ATMI C Function Reference*) and refuses the connection. If the configuration matches, the TEDG and the remote BEA TOP END system perform mutual authentication, using the protocol for the BEA TOP END Node Manager. If `SECURITY` is set to either `SAFE` or `PRIVATE` in the `DMCONFIG` file, the TEDG obtains an encryption key as part of the authentication process.

Encryption of messages between BEA TOP END and BEA Tuxedo systems is based on the BEA TOP END internode message security used between NIs. Internode message security is based, in turn, on the Kerberos 4.9 application libraries.

Note: To use BEA TOP END internode security, you must have the BEA TOP END Security Services Product installed on the same machine as the TEDG.

See Also

- "Configuring Security Between BEA TOP END and BEA Tuxedo Systems" on page 7-1

Part II Configuration

4 Configuring the TOP END Domain Gateway

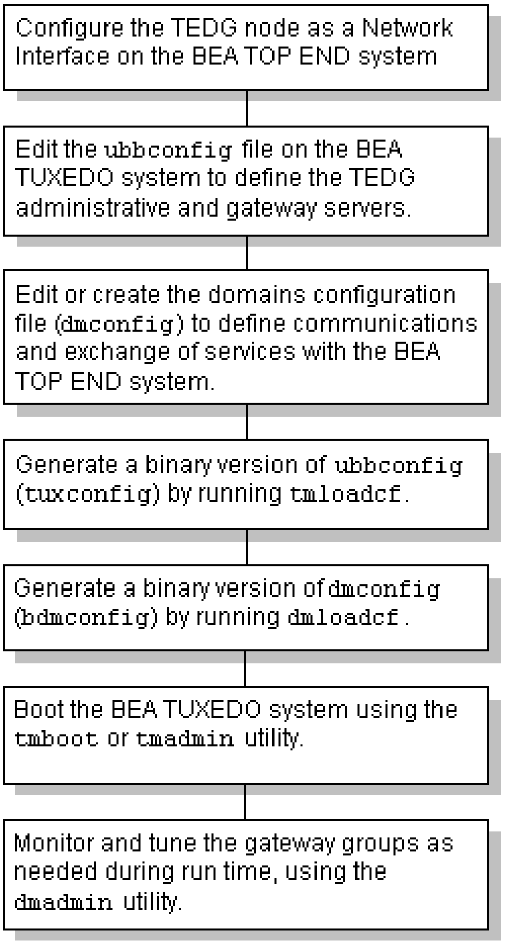
This topic includes the following sections:

- How to Configure the TEDG
- Rules for Configuring the TEDG
- Configuring a TEDG Node on the BEA TOP END System
- Communicating with BEA TOP END Systems Using Extended Node Names

How to Configure the TEDG

The procedure for configuring the TOP END Domain Gateway is shown in the following illustration.

Figure 4-1 TEDG Configuration and Run-Time Administration Process

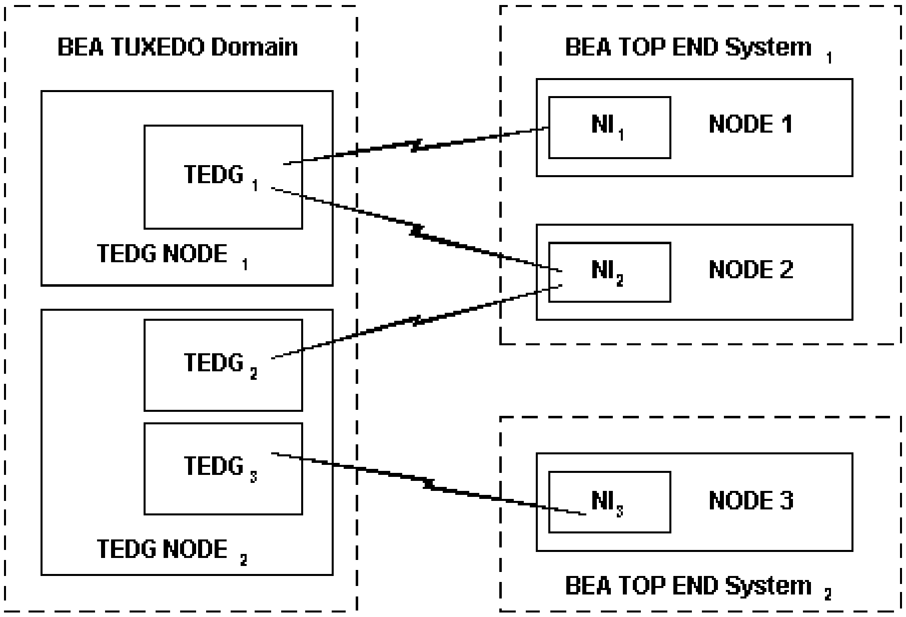

Configure the TEDG node as a Network Interface on the BEA TOP END system
Edit the ubbconfig file on the BEA TUXEDO system to define the TEDG administrative and gateway servers.
Edit or create the domains configuration file (dmconfig) to define communications and exchange of services with the BEA TOP END system.
Generate a binary version of ubbconfig (tuxconfig) by running tmloadcf.
Generate a binary version of dmconfig (bdmconfig) by running dmloadcf .
Boot the BEA TUXEDO system using the tmboot or tmadmin utility.
Monitor and tune the gateway groups as needed during run time, using the dmadmin utility.

Rules for Configuring the TEDG

You must adhere to the following rules when configuring the TEDG.

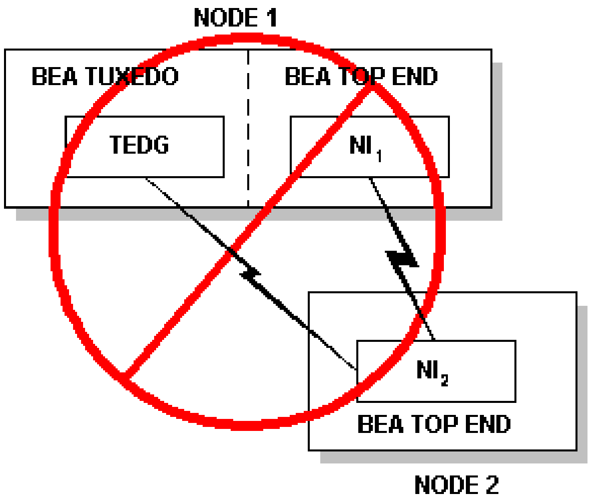
- A single copy of the TEDG can support simultaneous connections to multiple NI processes on different nodes of a single BEA TOP END system. Multiple copies of the TEDG can run within a BEA Tuxedo domain, even on a single node. This flexibility allows a given BEA Tuxedo domain to connect to more than one BEA TOP END system. It also enables the administrator to divide the connections to a BEA TOP END system among multiple TEDG processes for workload balancing and/or redundancy. The following illustration shows a valid TEDG configuration.

Figure 4-2 Valid TEDG Configuration



- BEA TOP END does not permit multiple network connections between any pair of nodes within a single BEA TOP END system, as shown in the following figure.

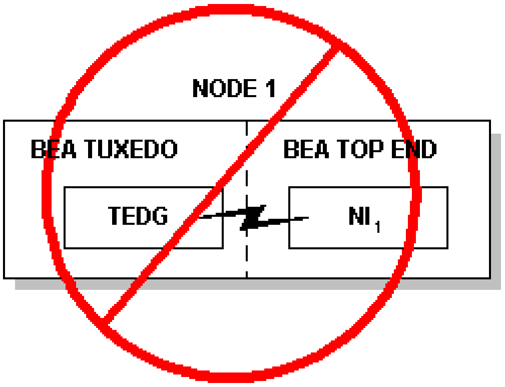
Figure 4-3 Invalid Multi-node TEDG Configuration



As you can see in the diagram, a single physical node, NODE 1, can support two logical nodes: the TEDG on the BEA Tuxedo system and NI_1 on the BEA TOP END system. In this scenario, the TEDG on NODE 1 is capable of functioning as an NI to NI_2 on NODE 2. The TEDG cannot be used to connect to NI_2 on NODE2, however, if another NI (in this case NI_1 on the BEA TOP END system on NODE 1) is connected to that node.

- Due to the way in which network connections and node names are managed in the BEA TOP END system, a BEA Tuxedo TEDG cannot communicate with a BEA TOP END system on the same node, as shown in the following figure. The TEDG and BEA TOP END NI are free to communicate with other BEA TOP END nodes as long as they do not communicate with the same node, as shown in the previous figure, “Invalid Multi-node TEDG Configuration.”

Figure 4-4 Invalid Single-node Configuration



- If multiple network adapters are used on a BEA TOP END node, the BEA TOP END nodemap file should be used to map the node names associated with the IP addresses of those network adapters to a single BEA TOP END node ID for the machine.

See Also

- "Communicating with BEA TOP END Systems Using Extended Node Names" on page 4-6

Configuring a TEDG Node on the BEA TOP END System

Use the BEA TOP END Interactive System Definition (ISD) software to add a TEDG node to the BEA TOP END system and configure an NI component to communicate with that node. For details on configuring an NI component, refer to the *BEA TOP END Interactive System Definition and Generation* manual.

Communicating with BEA TOP END Systems Using Extended Node Names

BEA TOP END Release 3.0 provides a feature called Extended Node Names that allows the use of external node names that are longer than eight characters. With the Extended Node Name feature, node names can be up to 255 characters long, while the internal node ID is an eight-byte opaque data structure used within the BEA TOP END system to identify nodes.

The mapping between node IDs and node names is established through a simple text file called the nodemap file. For details on creating the nodemap file, see *Getting Started with BEA TOP END*. The format of the file is described in the BEA TOP END `nodemap(4T)` reference page.

On BEA TOP END nodes, the nodemap file is installed in the following directories.

Platform	**Directory**
UNIX	`$TOPENDDIR/etc`
Windows 2000	`%TOPENDDIR%\etc`

If you have configured the TEDG to communicate with BEA TOP END nodes that have node names longer than eight characters, the nodemap file must be available to the TEDG. Before running the TEDG, copy the nodemap file from the BEA TOP END system to the following directories on the BEA Tuxedo system.

Platform	**Directory**
UNIX	`$TUXDIR/udataobj`
Windows 2000	`%TUXDIR%\udataobj`

The nodemap file also allows the administrator to map multiple node names to a BEA TOP END node identifier. This helps the TEDG, at incoming connection time, obtain a consistent node identifier for the NI running on a machine on which multiple network cards are installed.

See Also

- *Getting Started with BEA TOP END*
- `nodemap(4T)` in the *BEA TOP END Programmer's Reference Manual*

CHAPTER

5 Editing the UBBCONFIG File

This topic includes the following sections:

- What Is a UBBCONFIG File?
- Editing the UBBCONFIG File
- Sample Section of a UBBCONFIG File

What Is a UBBCONFIG File?

The `UBBCONFIG` file is a text version of the configuration file that defines a BEA Tuxedo application. You can create and edit a `UBBCONFIG` file with any text editor.

The `TUXCONFIG` file is a binary version of `UBBCONFIG`. It contains information used by `tmboot(1)` to start the servers and initialize the bulletin board of a BEA Tuxedo application in an orderly sequence. A `TUXCONFIG` file is created by executing the `tmloadcf(1)` command on the `UBBCONFIG` file.

See Also

- `UBBCONFIG(5)` in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*

- `tmboot(1)` and `tmloadcf(1)` in the *BEA Tuxedo Command Reference*

Editing the UBBCONFIG File

To enable connectivity between the BEA Tuxedo and BEA TOP END systems via the TEDG, you need to add the following two server groups to the `SERVERS` section of the `UBBCONFIG` file:

- The first "group" contains only one member: the domains administrative server (`DMADM`).
- The second group consists of the gateway administrative server (`GWADM`) and the gateway process that provides connectivity between a BEA Tuxedo domain and a BEA TOP END system (`GWTOPEND`). The `GWTOPEND` process must be in the same group as the `GWADM(5)` server, and the `GWADM` must be listed first.

These servers are defined as follows.

`DMADM`

The domains administrative server enables run-time administration of the configuration information required by domain gateway groups. The main function of this server is to provide run-time administration of the binary domains configuration file and to support a list of registered gateway groups. A domain may have no more than one instance of `DMADM`. If your BEA Tuxedo configuration already has a `DMADM` server for BEA Tuxedo interdomain communication, you do not need to supply a new one.

`GWADM`

The gateway administrative server enables run-time administration of a particular domain gateway group. There is one `GWADM` per gateway group. The main function of this server is to get domain configuration information from the `DMADM` server. It also provides administrative functionality and transaction logging (specifically, one log per gateway group).

`GWTOPEND`

This gateway process communicates with the Network Interface (NI) component on one or more nodes of a single BEA TOP END system. Different `GWTOPEND` gateways (in different BEA Tuxedo groups) may be configured to access different BEA TOP END systems or to split the load.

You cannot have multiple GWTOPEND processes in a group; you must add multiple pairs of GWADM and GWTOPEND processes based on the server group.

The gateways should not have reply queues (set REPLYQ = N) and should be marked as restartable (set RESTART=Y).

Also, specify a type of application security for your BEA Tuxedo domain using the SECURITY parameter in the RESOURCES section of the UBBCONFIG file.

For the syntax required for entries in the file, see UBBCONFIG(5) and GWTOPEND(5) in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*.

See Also

- DMADM(5) in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*
- GWADM(5) in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*
- GWTOPEND(5) in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*

Sample Section of a UBBCONFIG File

The following sample shows the GROUPS and SERVERS sections of a UBBCONFIG file modified for the TOP END Domain Gateway.

```
*GROUPS
DMADMGRP LMID=mach1 GRPNO=1
gwgrp LMID=mach1 GRPNO=2

*SERVERS
#GWTOPEND is the name of the TEDG binary program.

DMADM      SRVGRP="DMADMGRP" SRVID=1001 REPLYQ=N RESTART=Y
           MAXGEN=5 GRACE=3600
GWADM      SRVGRP="gwgrp" SRVID=1002 REPLYQ=N RESTART=Y MAXGEN=5
           GRACE=3600
```

```
GWTOPEND  SRVGRP="gwgrp" SRVID=1003 RQADDR="gwgrp" REPLYQ=N
          RESTART=Y MAXGEN=5 GRACE=3600
```

See Also

- `UBBCONFIG(5)` in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*

CHAPTER

6 Editing the DMCONFIG File

This topic includes the following sections:

- What Is a DMCONFIG File?
- Editing the DM_LOCAL_DOMAINS Section
- Editing the DM_REMOTE_DOMAINS Section
- Adding the DM_TOPEND Section
- Editing the DM_LOCAL_SERVICES Section
- Editing the DM_REMOTE_SERVICES Section
- Optional DMCONFIG Sections

See Also

- “Defining Security in the DMCONFIG File” on page 7-4
- “Configuring Multiple GWTOPEND Processes” on page 8-1
- “Sample Configuration File for a TOP END Domain Gateway” on page 9-1

What Is a DMCONFIG File?

A Domains configuration for the TEDG consists of at least one local BEA Tuxedo domain and at least one BEA TOP END system that can communicate and share services with the help of the BEA Tuxedo Domains feature. In a Domains configuration using the TEDG, a BEA TOP END system should be considered a "remote domain." How a BEA Tuxedo domain and a BEA TOP END system (remote domain) are connected and which services they make accessible to each other are defined in a Domains configuration file for the TEDG domain and the BEA TOP END system definition on the BEA TOP END system. The text version of a Domains configuration file is known as the `DMCONFIG` file after the environment variable used to hold the name of the actual file used.

A `DMCONFIG` file defines the following:

- The remote domains with which the local domain can communicate
- The local resources (such as services and queues) accessible to remote domains
- The remote resources accessible to the local domain
- Which local and remote resources are accessible through which gateways

The `DMCONFIG` file is parsed and loaded into a binary version, called `BDMCONFIG`, by the `dmloadcf(1)` utility. The `dmadmin(1)` command uses `BDMCONFIG` (or a copy of it) for monitoring the run-time application.

One `BDMCONFIG` file is required on each domain in a multi-domain configuration in which the Domains feature is being used.

The `DMCONFIG` and `BDMCONFIG` files are analogous to the `UBBCONFIG` and `TUXCONFIG` files used to define a BEA Tuxedo application.

A `DMCONFIG` file for a TEDG must contain five required sections and may contain one or two optional sections. The required sections are:

- `DM_LOCAL_DOMAINS`
- `DM_REMOTE_DOMAINS`
- `DM_TOPEND`
- `DM_LOCAL_SERVICES`

- `DM_REMOTE_SERVICES`

The optional sections are:

- `DM_ROUTING`
- `DM_ACCESS_CONTROL`

Domains Terminology Improvements

In this release, some of the Domains terminology is changing. The Domains MIB uses improved class and attribute terminology to describe the interaction between local and remote domains. While this improved terminology is more accurate than previous domains terminology, the scope of changes to domains-related documentation and error messages is limited in this release. The improved terminology has been applied to the `DM_MIB` classes, reference page, and error messages, the `DMCONFIG` file syntax, and various `DMCONFIG` error messages.

For backwards compatibility, aliases are provided between the `DMCONFIG` terminology used prior to this release and the improved Domains MIB terminology. In this release, `DMCONFIG` accepts both versions of the terminology. For details, see “Domains Terminology Improvements” on page -126 in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*.

See Also

- `DMCONFIG for GWTOPEND(5)` in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*
- `dmloadcf(1)` in the *BEA Tuxedo Command Reference*
- “Sample Configuration File for a TOP END Domain Gateway” on page 9-1

Editing the DM_LOCAL_DOMAINS Section

This section describes the environment required for a particular domain gateway group. Here a logical application name, *LDOM*, is assigned to the subset of local services that can be accessed by remote domains. Multiple entries in this section are used to define multiple gateway groups within a single BEA Tuxedo application. Each entry specifies the parameters required for the Domains gateway running in that group.

Entries in this section have the form:

LDOM required_parameters [*optional_parameters*]

where *LDOM* is an *identifier* value used to name the local domain. For the TEDG, each *LDOM* defines a gateway process that is part of a single BEA TOP END system. The *LDOM* communicates with remote domains of type TOPEND that are part of the same BEA TOP END system. (The BEA TOP END system name is defined in the DM_TOPEND section of DMCONFIG.)

For a description of the syntax required for parameters specified in this section, see DMCONFIG for GWTOPEND(5) in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*.

Defining the Required Parameters

The parameters required in this section are listed in the following table.

Table 6-1 Required Parameters in DM_LOCAL_DOMAINS

Required Parameter	Definition
GWGRP	Specifies the name of the gateway server group (as defined in the UBBCONFIG file) representing this local domain. There is a one-to-one relationship between a DOMAINID and the name of the gateway server group.
TYPE	Used for grouping local domains into classes. Valid entries are TDOMAIN, SNAX, OSITP, and TOPEND. For the TEDG, set this value to TOPEND.

Table 6-1 Required Parameters in DM_LOCAL_DOMAINS (Continued)

Required Parameter	Definition
DOMAINID	Used to identify the local domain. The TEDG uses the local domain DOMAINID as the BEA TOP END user ID for requests made to the BEA TOP END system. Define the associated password using the dmadmin subcommand topendpasswd. Refer to “Using the dmadmin Command Interpreter” on page 10-3 for information on using the dmadmin command. The DOMAINID must be unique across both local and remote domains.

Defining the Optional Parameters

Optional parameters describe the resources and limits used in the operation of domain gateways. These parameters, which are defined in detail in the DMCONFIG for GWTOPEND(5) reference page, are listed in the following table.

Table 6-2 Optional Parameters in DM_LOCAL_DOMAINS

This Parameter. . .	Specifies . . .
AUDITLOG	The name of the audit log for this local domain.
BLOCKTIME	The maximum wait time allowed for blocking a call.
CONNECTION_POLICY	The conditions under which a local domain gateway tries to establish a connection to a remote domain. Details on this parameter are provided in “Establishing Connections Between BEA Tuxedo and BEA TOP END Systems” on page 2-5.
DMTLOGDEV	The BEA Tuxedo file system that contains the Domains transaction log (DMTLOG) for this machine.
DMTLOGNAME	The name of the Domains transaction log for this domain.
DMTLOGSIZE	The numeric size, in pages, of the Domains transaction log for this machine.

Table 6-2 Optional Parameters in DM_LOCAL_DOMAINS (Continued)

This Parameter. . .	Specifies . . .
`MAXRDTRAN`	The maximum number of domains that can be involved in a transaction.
`MAXRETRY`	The maximum number of times that a domain gateway tries to establish a connection to a remote domain before quitting.
`MAXTRAN`	The maximum number of simultaneous global transactions allowed on this local domain.
`RETRY_INTERVAL`	The number of seconds between automatic attempts (all network addresses tried) to establish a connection to a remote domain.
`SECURITY`	The type of security to be used by the TEDG. Details on this parameter are provided in "Defining Security in the DMCONFIG File" on page 7-4.

See Also

- "Sample Configuration File for a TOP END Domain Gateway" on page 9-1
- `DMCONFIG for GWTOPEND(5)` in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*

Editing the DM_REMOTE_DOMAINS Section

This section identifies the remote domains that can be accessed by clients and servers of this Domains configuration. Entries in this section have the following form.

```
RDOM required_parameters
```

where *`RDOM`* is an identifier value used to name a remote domain to which a TEDG *`LDOM`* (as defined in the `DM_LOCAL_DOMAINS` section) may have a connection. The *`LDOM`* communicates with remote domains of type `TOPEND` that are part of the same BEA TOP END system as the *`LDOM`*. The BEA TOP END system name is defined in the `DM_TOPEND` section of `DMCONFIG`.

Note: Because of the BEA TOP END adjacent node routing topology, the services for the BEA TOP END system may reside on several different nodes. Therefore, a TEDG *`LDOM`* may need several *`RDOM`* entries to define connections to the BEA TOP END nodes where the desired BEA TOP END services reside.

For a description of the syntax required for parameters specified in this section, see `DMCONFIG for GWTOPEND(5)` in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*.

Defining the Required Parameters

The parameters required in the `DM_REMOTE_DOMAINS` section are listed in the following table.

Table 6-3 Required Parameters in DM_REMOTE_DOMAINS

Required Parameter	Definition
`TYPE`	Used for grouping remote domains into classes. `TYPE` can be set to one of the following values: `TOPEND`, `TDOMAIN`, `SNAX`, or `OSITP`. For the TEDG, set this value to `TOPEND`.
`DOMAINID`	Used to identify a remote domain. The TEDG uses the remote domain `DOMAINID` as the BEA Tuxedo user ID for requests made to the BEA Tuxedo system by the BEA TOP END system on this remote domain. `DOMAINID` must be unique across remote domains.

See Also

- “Sample Configuration File for a TOP END Domain Gateway” on page 9-1
- `DMCONFIG for GWTOPEND(5)` in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*

Adding the DM_TOPEND Section

The `DM_TOPEND` section defines:

- The network addressing information required by domains of `TYPE=TOPEND`
- The BEA TOP END system name for each BEA TOP END local domain (*`LDOM`*) and remote domain (*`RDOM`*) defined in the `DM_LOCAL_DOMAINS` and `DM_REMOTE_DOMAINS` sections

Defining the Required Parameters

The parameters required in this section are listed in the following table.

Table 6-4 Required Parameters in DM_TOPEND

Required Parameter	Definition
NWADDR	Specifies the network address associated with a local domain or remote domain. ■ If you associate the network address with an *LDOM*, the NWADDR parameter is used to accept connections from BEA TOP END systems. ■ If you associate the network address with an *RDOM*, the NWADDR parameter is used to initiate a connection. ■ If you associate multiple network addresses with an *RDOM*, the TP_SYSTEM value must be the same on each entry for the *RDOM*. ■ Multiple entries for an *RDOM* define primary and alternate network connections to the same physical node. The number of network addresses for an *RDOM* is not limited. When the TEDG attempts to make a connection, each network address is tried serially. Configuring too many network addresses or addresses that may not be operational can reduce TEDG performance. Multiple network addresses are not allowed for an *LDOM*. **Note:** Care should be taken when specifying the host address portion of the NWADDR parameter. When a BEA TOP END NI accepts a connection request that was issued from a TEDG, it resolves the network address of the TEDG to a name. The resolved name must match the defined hostname of the TEDG. If the defined hostname of the TEDG and the resolved name differ, including case, the NI connection fails. Such a failure may not be evident from either the GWTOPEND log file or the remote BEA TOP END NI log file. As a general rule, ensure that the hostname definitions match in the DMCONFIG file, the TOP END NI configuration file, the TOP END nodemap file, the TOP END tp_alias file, and the locally configured name resolution facilities. For further information on NI name resolution, refer to the tp_alias(4T) reference page in the *BEA TOP END Programmer's Reference Manual*.
TP_SYSTEM	Specifies the BEA TOP END system associated with the *LDOM* or *RDOM* defined in the DM_LOCAL_DOMAINS and DM_REMOTE_DOMAINS sections. The parameter accepts a string that corresponds to the BEA TOP END system name. The value of the string must match the value of the TP_SYSTEM environment variable specified in the node manager startup script. This script is described in the nm_script(4T) reference page.

Note: Do not configure *LDOM*s, *RDOM*s, and their network addresses such that more than one TEDG connection between a BEA TEDG node and a BEA TOP END node is activated for a particular TP_SYSTEM name. BEA TOP END network interface protocol does not support multiple connections of this type. As a result, if more than one TEDG connection is active, the TEDG or the BEA TOP END node rejects the duplicate connections. Due to variations in how network addresses can be specified, this type of configuration cannot be fully validated in this configuration file.

See Also

- "Sample Configuration File for a TOP END Domain Gateway" on page 9-1
- DMCONFIG for GWTOPEND(5) in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*
- nm_script(4T) reference page in the *BEA TOP END Programmer's Reference Manual*

Editing the DM_LOCAL_SERVICES Section

The DM_LOCAL_SERVICES section defines the mapping required to make BEA Tuxedo services and /Q queue spaces available to a BEA TOP END system.

In DMCONFIG files written for domain gateways other than the TEDG, the purpose of entries in the DM_LOCAL_SERVICES section is to map local services to remote names for those services. In a DMCONFIG file written for the TEDG, however, the purpose of entries in this section is to map BEA Tuxedo names to BEA TOP END names for the following:

- Request/reply and conversational services
- BEA Tuxedo queue spaces and queue names

For BEA TOP END and BEA Tuxedo systems to exchange queue messages, there must be a mapping of the queue addresses on both systems. The address of a BEA Tuxedo /Q queue consists of a combination of queue space (`QSPACE`) and queue name (`QNAME`).

Defining Service and Queue Mapping Entries

Entries in this section must adhere to the following form.

```
service [TYPE=SERVICE] required_parameters [optional_parameters]
qspace  TYPE=QSPACE    required_parameters [optional_parameters]
qname   TYPE=QNAME     required_parameters [optional_parameters]
```

Here:

- *`service`* is the name of an exported BEA Tuxedo service (`TYPE=SERVICE` is the default).
- *`qspace`* is the name of an exported BEA Tuxedo queue space.
- *`qname`* is the name of a queue defined in a BEA Tuxedo queue space and used in `tpenqueue(3c)` requests.

Configuration Guidelines for DM_LOCAL_SERVICES

Observe the following guidelines when editing the `DM_LOCAL_SERVICES` section of the `DMCONFIG` file.

- For `SERVICE` and `QSPACE` entries, the BEA Tuxedo service or queue space is assumed to be available in the local BEA Tuxedo domain. A queue name specified in a `QNAME` entry is assumed to be defined in the queue space associated with that queue.
- If the configuration includes `LDOM`s for more than one BEA TOP END system, or if it includes multiple gateway types, specify the `LDOM` parameter in the local service entry. Do not specify a mixed configuration that does not specify an `LDOM;` such a configuration may prevent a gateway from initializing properly. If in doubt, explicitly set `LDOM`.

- Entries configured for a specific LDOM are advertised by the gateway for that LDOM. Additionally, any local SERVICE entry that is not configured for a particular local domain using the LDOM parameter, and that has the required TE_PRODUCT parameter, is advertised to the BEA TOP END system by each local domain of type TOPEND.
- Because SERVICE and QSPACE entries configure BEA TOP END service identifiers that are advertised as BEA TOP END services, these identifiers must not overlap for a particular LDOM. For example, do not specify both a SERVICE entry and a QSPACE entry such that the TE_PRODUCT and TE_FUNCTION parameters of the SERVICE entry match the TE_RTQGROUP and TE_RTQNAME of the QSPACE entry, respectively.
- QSPACE and QNAME entries are independent. Any combination of QSPACE and QNAME entries may be used by an application by supplying the associated BEA TOP END identifiers in the tp_rtq_put(3T) routine call. A run-time error will occur if the specified combination does not exist in the local BEA Tuxedo domain.

Defining Service Type Parameters

A SERVICE entry defines BEA Tuxedo services that are accessible from a BEA TOP END system via the TEDG. These services are made accessible by mapping the BEA TOP END service identifiers to BEA Tuxedo service names. These service identifiers are used with the BEA TOP END tp_client_send(3T) and tp_client_signon(3T) routine calls. RTQ service requests are not mapped in SERVICE entries.

The following table lists the parameters available for mapping in entries of type SERVICE in the DM_LOCAL_SERVICES section. Some parameters are required; others are optional.

Table 6-5 Parameters of SERVICE Type Entries in DM_LOCAL_SERVICES

This Parameter . . .	Defines . . .	Required?
TE_PRODUCT	The BEA TOP END product name.	Yes
TE_FUNCTION	The BEA TOP END function name.	Yes
TE_TARGET	The BEA TOP END MSR target.	No

Table 6-5 Parameters of SERVICE Type Entries in DM_LOCAL_SERVICES

This Parameter . . .	Defines . . .	Required?
`TE_QUALIFIER`	The BEA TOP END function qualifier.	No
`TYPE`	A type for this entry. The value `SERVICE` means that the purpose of this entry is to map various parameters for a local BEA Tuxedo service being exported to a BEA TOP END system.	No
`ACL`	The name of the access control list (ACL) to be used by the TEDG to restrict requests made to the specified `SERVICE` by BEA TOP END systems. You define the ACL in the `DM_ACCESS_CONTROL` section of `DMCONFIG`, as described in “Defining Security in the DMCONFIG File” on page 7-4.	No
`LDOM`	The name of the local domain exporting the specified service. If you do not specify this parameter, then the entry applies to all the local domains of type `TOPEND` that are defined in the `DM_LOCAL_DOMAINS` section.	No
`INBUFTYPE`	The input buffer type allowed for this service. You can restrict the input buffer type by setting one of the following values: `FML32`, `CARRAY`, or `X_OCTET`.	No
`OUTBUFTYPE`	The output buffer type accepted from this service. You can restrict this to one of the following buffer types: `FML32`, `CARRAY`, or `X_OCTET`.	No

Defining QSPACE Type Parameters

A `QSPACE` entry defines a BEA Tuxedo queue space that is made available to the BEA TOP END system by the TEDG as if it were an RTQ queue (limitations apply). RTQ queues are made available in a BEA TOP END system by advertising the RTQ Group and Queue name as a BEA TOP END service name. In a manner similar to that used by the RTQ server, the TEDG handles `tp_rtq_put(3T)` requests sent to its RTQ queues. Each request is mapped to the BEA Tuxedo queue space identified in this `QSPACE` entry. Both `QSPACE` entries and `QNAME` entries are required for message queuing.

The following table lists the parameters available for mapping entries of type `QSPACE` in the `DM_LOCAL_SERVICES` section. Some parameters are required; others are optional.

Table 6-6 Parameters of QSPACE Type Entries in DM_LOCAL_SERVICES

This Parameter . . .	Defines . . .	Required?
`TYPE`	A type for this entry. The value `QSPACE` means that the purpose of this entry is to map various parameters to a local BEA Tuxedo queue space being made available to a BEA TOP END system as an RTQ queue.	Yes
`TE_RTQGROUP`	The BEA TOP END RTQ group name.	Yes
`TE_RTQNAME`	The BEA TOP END RTQ queue name.	Yes
`TE_TARGET`	The BEA TOP END MSR target.	No
`ACL`	The name of the access control list (ACL) to be used by the TEDG to restrict requests made to the specified `QSPACE` by BEA TOP END systems. You define the ACL in the `DM_ACCESS_CONTROL` section of `DMCONFIG` as described in "Defining Security in the DMCONFIG File" on page 7-4.	No
`LDOM`	The name of the local domain exporting the specified queue space. If you do not specify this parameter, then the entry applies to all the local domains of type `TOPEND` that are defined in the `DM_LOCAL_DOMAINS` section.	No

Defining QNAME Type Parameters

A `QNAME` entry maps BEA TOP END service names to BEA Tuxedo queue names for requests enqueued to a BEA Tuxedo application via RTQ. The services defined in `QNAME` entries are not advertised as services to a BEA TOP END system.

The following table lists the parameters available for mapping entries of type `QNAME` in the `DM_LOCAL_SERVICES` section. Some parameters are required; others are optional.

Table 6-7 Parameters of QNAME Type Entries in DM_LOCAL_SERVICES

This Parameter . . .	Defines . . .	Required?
`TYPE`	A type for this entry. The value `QNAME` means that the purpose of this entry is to define the parameters used in mapping a BEA TOP END service name to a BEA Tuxedo queue name for requests enqueued to a BEA Tuxedo application via RTQ.	Yes
`TE_PRODUCT`	The BEA TOP END product name.	Yes
`TE_FUNCTION`	The BEA TOP END function name.	Yes
`TE_TARGET`	The BEA TOP END MSR target.	No
`TE_QUALIFIER`	The BEA TOP END function qualifier.	No
`LDOM`	The name of the local domain to which this `QNAME` entry applies. If you do not specify this parameter, then the entry applies to all the local domains of type `TOPEND` that are defined in the `DM_LOCAL_DOMAINS` section.	No

See Also

- "Sample Configuration File for a TOP END Domain Gateway" on page 9-1
- `DMCONFIG for GWTOPEND(5)` in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*
- `tpenqueue(3c)` in the *BEA Tuxedo ATMI C Function Reference*

Editing the DM_REMOTE_SERVICES Section

The `DM_REMOTE_SERVICES` section defines the mapping information required to make BEA TOP END services, RTQ queues, and services accessed via RTQ available to a BEA Tuxedo application. This section is required for TOP END Domain Gateways.

In `DMCONFIG` files written for domain gateways other than the TEDG, the purpose of entries in the `DM_REMOTE_SERVICES` section is to map local services to remote names for those services. In a `DMCONFIG` file written for the TEDG, however, the purpose of entries in this section is to map:

- Request/reply and conversational services
- BEA Tuxedo queue spaces and queue names

For BEA Tuxedo and BEA TOP END systems to exchange queue messages, there must be a mapping of the queue addresses on both systems. The address of a BEA Tuxedo /Q queue consists of a combination of queue space (`QSPACE`) and queue name (`QNAME`).

Defining Service and Queue Mapping Entries

The entries in this section must adhere to the following form.

```
service [TYPE=SERVICE] required_parameters [optional_parameters]
qspace  TYPE=QSPACE    required_parameters [optional_parameters]
qname   TYPE=QNAME     required_parameters [optional_parameters]
```

Here:

- *`service`* is the BEA Tuxedo service assigned to the BEA TOP END service (default).
- *`qspace`* is the name of the BEA Tuxedo queue space assigned to the RTQ queue.
- *`qname`* is the name of the BEA Tuxedo queue assigned to a BEA TOP END service accessed through RTQ.

Configuration Guidelines for the DM_REMOTE_SERVICES

Observe the following guidelines when editing the `DM_REMOTE_SERVICES` section of the `DMCONFIG` file.

- Because `SERVICE` and `QSPACE` entries define service identifiers and `qspace` identifiers that are advertised as BEA Tuxedo services, these identifiers must be unique for a particular `LDOM`. However, multiple entries of the same type and identifier are permitted for load balancing. All entries for a single service identifier must have the same value for the `CONV` parameter.
- If they do not define a particular local domain (with the `LDOM` parameter), the following types of entries apply to each local domain of type `TOPEND`:
  - Any `SERVICE` and `QNAME` entry that includes the `TE_PRODUCT` parameter
  - Any `QSPACE` entry that includes the `TE_RTQGROUP` parameter
- For `SERVICE` and `QSPACE` entries, the BEA TOP END service or RTQ queue is assumed to be available on the remote BEA TOP END node specified with the `RDOM` parameter or obtained through specified routing criteria. Services specified in a `QNAME` entry are assumed to be available via the BEA TOP END node to which they are enqueued.
- Entries in which multiple remote domains are specified with the `RDOM` parameter use the Domains Failover feature that enables this service request to be routed to the alternate `RDOM` if the primary `RDOM` is not connected, or to the second alternate if the primary and first alternate are not connected. This feature may also be used in conjunction with the load balancing feature in which multiple remote service entries are configured for the same service or queue space identifier.
- If a configuration includes `LDOM`s for more than one BEA TOP END system, or includes multiple gateway types (such as `TDOMAIN` and `TOPEND`), the `LDOM` parameter should be specified in the remote service entry. If an `RDOM` is specified on the entry or in a referenced routing entry, it should match the `LDOM` type (`TOPEND`) and `TP_SYSTEM`. Mixed configurations that do not specify `LDOM`, or reference `RDOM`s of mixed types or mixed `TP_SYSTEM`s should not be created and may prevent a gateway from initializing properly. If in doubt, explicitly set `LDOM` and a remote domain (via the `RDOM` parameter or `ROUTING`). "Wildcard"

specifications for remote domains should be used only when a single gateway type is defined.

Defining the Service Type Parameters

A `SERVICE` entry defines BEA TOP END services that are accessible from a BEA Tuxedo domain via the TEDG. These BEA TOP END services are made accessible by mapping BEA Tuxedo service names to BEA TOP END service identifiers. These service names are used with two BEA Tuxedo functions: `tpcall(3c)` and `tpacall(3c)`.

The following table lists the parameters available for mapping entries of type `SERVICE` in the `DM_REMOTE_SERVICES` section. Some parameters are required; others are optional.

Table 6-8 Parameters of SERVICE Type Entries in DM_REMOTE_SERVICES

This Parameter . . .	Defines . . .	Required?
`TE_PRODUCT`	The BEA TOP END product name.	Yes
`TE_FUNCTION`	The BEA TOP END function name.	Yes
`TE_TARGET`	The BEA TOP END MSR target.	No
`TE_QUALIFIER`	The BEA TOP END function qualifier.	No
`TYPE`	A type for this entry. The value `SERVICE` means that the purpose of this entry is to define the parameters available when mapping a BEA TOP END service to a local BEA Tuxedo service.	No
`LDOM`	The name of the local domain importing this service. If you do not specify this parameter, then the entry applies to all the local domains of type `TOPEND` that are defined in the `DM_LOCAL_DOMAINS` section.	No

Table 6-8 Parameters of SERVICE Type Entries in DM_REMOTE_SERVICES (Continued)

This Parameter . . .	Defines . . .	Required?
`RDOM`	The name of the remote domain providing this service. The remote domain must be of type `TOPEND` and it must belong to the `TP_SYSTEM` to which the local domain (that is, the domain to which this entry applies) belongs. If you do not specify the `RDOM` parameter and routing criteria, the local domain assumes that any remote domain of type `TOPEND` with the same `TP_SYSTEM` value as the local domain provides the service. To configure alternate remote domains using this parameter, you must specify `ON_STARTUP` as the value of the `CONNECTION_POLICY` parameter in the `DM_LOCAL_DOMAINS` section.	No
`INBUFTYPE`	The input buffer type allowed for this service. You can restrict the input buffer type by setting one of the following values: `FML32`, `CARRAY`, or `X_OCTET`.	No
`OUTBUFTYPE`	The output buffer type accepted from this service. You can restrict this to one of the following buffer types: `FML32`, `CARRAY`, or `X_OCTET`.	No
`CONV`	Request/Response or conversational service mapping. If this value is set to `Y`, the BEA TOP END server application must manage a pseudo-conversation that may or may not maintain application context. The default is `N`.	No
`TRANTIME`	The default timeout value, in seconds, for a transaction automatically started for the associated service.	No

Table 6-8 Parameters of SERVICE Type Entries in DM_REMOTE_SERVICES (Continued)

This Parameter . . .	Defines . . .	Required?
ROUTING	The name of the criteria used by a local domain for data-dependent routing when more than one remote domain offers the same service. If you do not specify this parameter, data-dependent routing is not done for this service.	No

Defining QSPACE Type Parameters

A QSPACE entry defines a BEA TOP END RTQ queue that is made available in the BEA Tuxedo domain by the TEDG as if it were a BEA Tuxedo queue space. Queue spaces are made available in a BEA Tuxedo application by advertising the queue space name as a BEA Tuxedo service name. In a manner similar to that used by the TMQUEUE server, the TEDG handles tpenqueue requests sent to its queue space names. Each request is mapped to the RTQ queue identified in this QSPACE entry. Both QSPACE entries and QNAME entries are required for message queuing.

The following table lists the parameters available for mapping entries of type QSPACE in the DM_REMOTE_SERVICES section. Some parameters are required; others are optional.

Table 6-9 Parameters of QSPACE Type Entries in DM_REMOTE_SERVICES

This Parameter . . .	Defines . . .	Required?
TYPE	A type for this entry. The value QSPACE means that the purpose of this entry is to define the parameters available when mapping a BEA TOP END RTQ queue to a local BEA Tuxedo queue space.	Yes
TE_RTQGROUP	The BEA TOP END RTQ group name.	Yes
TE_RTQNAME	The BEA TOP END RTQ queue name.	Yes
TE_TARGET	The BEA TOP END MSR target.	No

Table 6-9 Parameters of QSPACE Type Entries in DM_REMOTE_SERVICES

This Parameter . . .	Defines . . .	Required?
`LDOM`	The name of the local domain importing this RTQ queue. If you do not specify this parameter, then the entry applies to all the local domains of type `TOPEND` that are defined in the `DM_LOCAL_DOMAINS` section.	No
`RDOM`	The name of the remote domain providing this RTQ queue. The remote domain must be of type `TOPEND` and it must belong to the same `TP_SYSTEM` to which the local domain (that is, the domain to which this entry applies) belongs. If you do not specify the `RDOM` parameter and routing criteria, the local domain assumes that any remote domain of type `TOPEND` with the same `TP_SYSTEM` value as the local domain provides the service. To configure alternate remote domains using this parameter, you must specify `ON_STARTUP` as the value of the `CONNECTION_POLICY` parameter in the `DM_LOCAL_DOMAINS` section.	No
`TRANTIME`	The default timeout value, in seconds, for a transaction automatically started for the associated service.	No
`ROUTING`	The name of the criteria used by a local domain for data-dependent routing when more than one remote domain offers the same service. If you do not specify this parameter, data-dependent routing is not done for this service.	No

Defining QNAME Type Parameters

A `QNAME` entry maps a BEA Tuxedo queue name to a BEA TOP END service (product and function) for requests enqueued to a BEA TOP END system. `QNAME` entries are not advertised as services to a BEA Tuxedo application.

The following table lists the parameters available for mapping entries of type QNAME in the DM_REMOTE_SERVICES section. Some parameters are required; others are optional.

Table 6-10 Parameters of QNAME Type Entries in DM_REMOTE_SERVICES

This Parameter . . .	Defines . . .	Required?
TYPE	A type for this entry. The value QNAME means that the purpose of this entry is to define the parameters available when mapping a BEA Tuxedo queue name to a BEA TOP END service name for requests enqueued to a BEA TOP END system via /Q.	Yes
TE_PRODUCT	The BEA TOP END product name.	Yes
TE_FUNCTION	The BEA TOP END function name.	Yes
TE_TARGET	The BEA TOP END MSR target.	No
TE_QUALIFIER	The BEA TOP END function qualifier.	No
LDOM	The name of the local domain to which the queue name entry applies. If you do not specify this parameter, then the entry applies to all the local domains of type TOPEND that are defined in the DM_LOCAL_DOMAINS section.	No

See Also

- "Sample Configuration File for a TOP END Domain Gateway" on page 9-1
- tpacall(3c)in the *BEA Tuxedo ATMI C Function Reference*
- tpcall(3c) in the *BEA Tuxedo ATMI C Function Reference*
- DMCONFIG for GWTOPEND(5) in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*

Optional DMCONFIG Sections

A `DMCONFIG` file for a TEDG may include either or both of the following optional sections:

- `DM_ROUTING`

  This section provides information for data-dependent routing of service requests. Only `FML32` descriptions are applicable to gateways of type `TOPEND`. Refer to `DMCONFIG for GWTOPEND(5)` for details.

- `DM_ACCESS_CONTROL`

  This section specifies the access control lists used by the local domain. For additional information, refer to “Defining Security in the DMCONFIG File” on page 7-4 and the `DMCONFIG for GWTOPEND(5)` reference page in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*.

See Also

- “Sample Configuration File for a TOP END Domain Gateway” on page 9-1
- `DMCONFIG for GWTOPEND(5)` in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*

CHAPTER

7 Configuring Security Between BEA TOP END and BEA Tuxedo Systems

This topic includes the following sections:

- How Security Is Provided Between BEA Tuxedo and BEA TOP END Systems
- Security Prerequisites
- Configuring Security in a BEA Tuxedo System
- Using BEA Tuxedo Security Administration Tools to Authorize Intersystem Access
- Defining a BEA TOP END Password for the TEDG
- Using BEA TOP END Security Administration Tools to Authorize Intersystem Access
- Configuring TEDG-to-NI Encryption and Authentication

See Also

- "Administering Security" on page 2-1 in *Using Security in CORBA Applications*

- "Understanding Security Between the BEA TOP END and BEA Tuxedo Systems" on page 3-1

How Security Is Provided Between BEA Tuxedo and BEA TOP END Systems

Security is provided between BEA Tuxedo and BEA TOP END systems as follows.

For Requests Passed . . .	To . . .	Use . . .
From BEA Tuxedo clients	TOP END Domain Gateways (TEDG)	Normal BEA Tuxedo security methods
From the TEDG	BEA Tuxedo servers or queues	Normal BEA Tuxedo security methods
From BEA TOP END clients	TEDG	Normal BEA TOP END security methods
Through the TEDG	N/A	Parameters defined in the `DMCONFIG` file for the gateway

In addition, you have the following options:

- You can configure the TEDG to require authentication when connecting to specific BEA TOP END systems.
- You can encrypt the links between BEA TOP END and BEA Tuxedo systems.

Security Prerequisites

The BEA TOP END Security Services Product, version 3.0, is required for security between BEA TOP END and BEA Tuxedo systems. The product must be installed on all BEA TOP END nodes and on any BEA Tuxedo node running a TEDG that has been configured for security.

On the Windows 2000 platform, the BEA TOP END Base product is a prerequisite for installation of the BEA TOP END Security Services product. Therefore, both the BEA TOP END Base product and the BEA TOP END Security Services product must be installed on a Windows 2000 machine that is being used as a BEA Tuxedo node running a TEDG that has been configured for security. Under these circumstances, both products must be installed even if you will not be running a BEA TOP END application on the Windows 2000 machine.

Configuring Security in a BEA Tuxedo System

- Defining Security in the UBBCONFIG File
- Defining Security in the DMCONFIG File

Defining Security in the UBBCONFIG File

Use the `SECURITY` parameter in the `RESOURCES` section of the `UBBCONFIG` file to specify the type of application security for the BEA Tuxedo domain. This is applicable to the interaction between:

- BEA Tuxedo clients and the TEDG
- The TEDG and BEA Tuxedo servers/queues

For valid values and syntax for the SECURITY parameter, refer to the UBBCONFIG(5) reference page in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*.

Defining Security in the DMCONFIG File

The following sections in the DMCONFIG file contain security parameters you define to establish security for a configuration that includes the TEDG:

- DM_LOCAL_DOMAINS Section
- DM_ACCESS_CONTROL Section
- DM_LOCAL_SERVICES Section

DM_LOCAL_DOMAINS Section

The SECURITY parameter specified in the DM_LOCAL_DOMAINS section of the DMCONFIG file controls the security level for the TEDG. This parameter specifies whether BEA TOP END security is used by the TEDG for internode authentication and protection. If TYPE=TOPEND, then the following values are valid for the SECURITY parameter:

- NONE
- CLEAR
- SAFE
- PRIVATE

This Parameter . . .	Uses BEA TOP END Security	And Specifies . . .
NONE	No	The default value

This Parameter . . .	Uses BEA TOP END Security	And Specifies . . .
`CLEAR`	Yes	No protection is required for internode messages
`SAFE`	Yes	Messages should be sent using the Kerberos SAFE message checksum
`PRIVATE`	Yes	Messages should be encrypted using the Kerberos 4 implementation of DES

Values for the `SECURITY` parameter must be consistent with the BEA TOP END Node Manager (NM) configuration parameters [`security`] and [`internode security`] as described in `nm_config(4T)`. Consistency is checked during node signon.

DM_ACCESS_CONTROL Section

This optional section contains local Access Control Lists (`ACL`) used by the TEDG to restrict access by remote domains to local resources. Each entry consists of an `ACL_NAME` resource identifier along with a list of required parameters designating remote domains permitted to access the resource. If no entry exists for a local service, the service is accessible to all remote domains.

DM_LOCAL_SERVICES Section

The optional `ACL` parameter is used by the TEDG to restrict requests from a BEA TOP END remote domain made to specific services or queue spaces defined in `SERVICE` and `QSPACE` entries, respectively. Define the `ACL` parameter as follows:

`ACL =` *`identifier`*

where *`identifier`* specifies the name of the access control list (ACL) to be used by the TEDG to restrict requests made to the target service or queue space by BEA TOP END systems. 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 the service or queue space defined in this entry.

See Also

- `DMCONFIG for GWTOPEND(5)` in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*
- `nm_config(4T)` in the *BEA TOP END Programmer's Reference Manual*

Using BEA Tuxedo Security Administration Tools to Authorize Intersystem Access

To access BEA Tuxedo services, the TEDG uses the BEA Tuxedo user ID assigned, via `DOMAINID`, to the appropriate remote domain.

To establish access, by a BEA TOP END application, to BEA Tuxedo resources (services and queue spaces), complete the following procedure.

1. For each remote domain defined as type `TOPEND` in the `DMCONFIG` file, add an entry (remote domain `DOMAINID` and password) in the BEA Tuxedo security data files, `tpusr` and `tpgrp`, and assign the user ID entry to a group. To do so, enter the following command:

   ```
   tpusradd -u uid -g gid DOMAINID
   ```

   You will be prompted for a password for each user ID.

   If the application is not active, you must run `tpusradd` on the master node. If the application is active, you can run this command on any node.

   Note: You can add these entries to an existing group, or to a new group. New groups must be created before the `tpusradd` command can be used.To create a new group, use the `tpgrpadd` command. For the required syntax, see `tpgrpadd(1)` in the *BEA Tuxedo Command Reference*.

2. Define the `SECURITY` parameter in the `UBBCONFIG` file. Add ACL entries based on the following settings in the `UBBCONFIG` file:

   - If `SECURITY=ACL`, then you *may* include an entry in the BEA Tuxedo security file for each service and queue space to be accessed by a BEA TOP END remote domain. If you do, add the group associated with each BEA TOP END remote domain `DOMAINID` that will access a service or queue space to the entry in the BEA Tuxedo security file for that service or queue space.
   - If `SECURITY=MANDATORY_ACL`, then you *must* include an entry in the ACL database for each service and queue space to be accessed by any BEA TOP END remote domain. Add the group associated with the `DOMAINID` for each BEA TOP END remote domain that will access a service or queue space to the entry in the BEA Tuxedo security file for that service or queue space.

3. Run the `tpacladd(1)` command to add an ACL entry to the BEA Tuxedo security data files, thus authorizing access to BEA Tuxedo resources (that is, services and/or queue spaces) as needed, for each remote domain.

   The format of the `tpacladd` command is as follows:

   ```
   tpacladd -g gid servicename
   ```

   ```
   tpacladd -g gid queue_space
   ```

   Note: These commands authorize access to the specified service or queue space for the owners of all user IDs in the group.

See Also

- `tpacladd(1)` in the *BEA Tuxedo Command Reference*
- `tpusradd(1)` in the *BEA Tuxedo Command Reference*

Defining a BEA TOP END Password for the TEDG

To access BEA TOP END services through RTQ requests, the TEDG uses the BEA TOP END user ID assigned, via `DOMAINID`, to the local domain. For each local domain defined as type `TOPEND` in the `DMCONFIG` file, you must define a password for the BEA TOP END user ID. To define a password, start the `dmadmin(1)` utility and enter the `topendpasswd` command. (See "Using the dmadmin Command Interpreter" on page 10-3 for details.)

Note: Non-RTQ access to BEA TOP END services is granted by defining the TEDG nodes as part of the BEA TOP END system, listing the relevant remote services in the `DMCONFIG` file, and configuring BEA Tuxedo user access to the TEDG advertised services.

See Also

- `dmadmin(1)` in the *BEA Tuxedo Command Reference*

Using BEA TOP END Security Administration Tools to Authorize Intersystem Access

After each system generation on the BEA TOP END administration node, add the new BEA Tuxedo services to the BEA TOP END product and function lists. Updating these lists makes it possible to use the `tpsecure(1T)` utility to authorize BEA TOP END users to access BEA Tuxedo services and queues.

1. The file `$TOPENDADM/admin/$TP_SYSTEM/product.lst` contains a list of products defined for the BEA TOP END system and is used to provide choices from which the `tpsecure(1T)` user may select. If they are not included, add the following names to the list:

   a. The product name for each `SERVICE` entry in the `DM_LOCAL_SERVICES` section that specifies a `TE_PRODUCT` and `TE_FUNCTION` parameter.

   b. The RTQ group name for each `QSPACE` entry in the `DM_LOCAL_SERVICES` section that specifies a `TE_RTQGROUP` and `TE_RTQNAME` parameter.

2. The file `$TOPENDADM/admin/$TP_SYSTEM/`*`prodname`*`.fnc`, where *`prodname`* is the product name, contains a list of the functions defined for the product in the BEA TOP END system, and is used to provide choices from which the `tpsecure(1T)` user may select. If the list of functions is incomplete, update it as follows:

   a. Add the function name for each `SERVICE` entry in the `DM_LOCAL_SERVICES` section that specifies a `TE_PRODUCT` and `TE_FUNCTION` parameter to the list of functions for the corresponding product.

   b. Add the RTQ queue name (`TE_RTQNAME`) for each `QSPACE` entry in the `DM_LOCAL_SERVICES` section that specifies a `TE_RTQGROUP` and `TE_RTQNAME` parameter to the list of functions for the corresponding product (`TE_RTQGROUP`).

   Note: BEA TOP END security requires a UNIX administration node. Hence these files reside only on UNIX systems.

3. Use the BEA TOP END `tpsecure(1T)` utility on the BEA TOP END administration node to do the following:

   - Define each BEA TOP END user ID (equal to the local domain `DOMAINID`) and password created by `dmadmin(1)` to be used to communicate with the BEA TOP END system.
   - Authorize owners of BEA TOP END user IDs to access BEA TOP END resources (products/functions) as needed. This step is required only for accessing queues.
   - Authorize BEA TOP END users to access BEA Tuxedo resources via the newly defined products and functions.

See Also

- `tpsecure(1T)` *in the BEA TOP END Programmer's Reference Manual*
- *BEA TOP END Runtime Administration on UNIX*

- *BEA TOP END Security Services on UNIX*

Configuring TEDG-to-NI Encryption and Authentication

If, in the `DMCONFIG` file, you have assigned a value other than `NONE` to the `SECURITY` parameter, then you must establish mutual authentication and encryption. To do so, complete the following procedure.

1. In the BEA TOP END security database, define a Kerberos principle of the form *`node.system`* for each node (machine) running the TEDG. The value of *`node`* is the name of the machine; the value of *`system`* is the name of the BEA TOP END system. (On a UNIX system, you can obtain the name of the machine by running the `uname -n` command.)
2. Generate a Kerberos `SRVTAB` file for each node and make all such files available to each TEDG at start of day. These files are needed by the TEDG when a security level (`CLEAR`, `SAFE`, or `PRIVATE`) is configured in the `DMCONFIG` file.

   For each principle, create a `SRVTAB` file by using the `ext_srvtab(1T)` utility on the Kerberos master node. Rename each file as `srvtab.`*`system`*, where the value of *`system`* is the BEA TOP END system name of the principle.
3. Copy each file to the appropriate directory (defined by the `APPDIR` environment variable) on the TEDG node.

See Also

- `ext_srvtab(1T)` in the *BEA TOP END Programmer's Reference Manual*
- *BEA TOP END Runtime Administration on UNIX*
- *BEA TOP END Security Services on UNIX*

8 Communicating with Multiple BEA TOP END Systems

This topic includes the following sections:

- Configuring Multiple GWTOPEND Processes
- Defining Multiple Processes in the UBBCONFIG file
- Modifying the DMCONFIG File

See Also

- "Configuring the TOP END Domain Gateway" on page 4-1

Configuring Multiple GWTOPEND Processes

To communicate with multiple BEA TOP END systems, you need to configure multiple `GWTOPEND` processes. Although a single gateway instance of the TEDG may communicate with multiple BEA TOP END nodes, each of those nodes must belong to the same BEA TOP END system. Configuring multiple GWTOPEND processes provides the capability to communicate with multiple BEA TOP END systems. To configure multiple GWTOPEND processes:

1. Define the multiple gateway groups in the UBBCONFIG file.
2. Add the gateway groups to the DMCONFIG file.

Defining Multiple Processes in the UBBCONFIG file

To configure multiple GWTOPEND processes, you need to define multiple gateway groups. To do so, you must define multiple GWADM/GWTOPEND pairs in the UBBCONFIG file, and assign each pair to a different BEA Tuxedo group.

Sample UBBCONFIG File

The following example shows the GROUPS and SERVERS sections of the UBBCONFIG file configured for two BEA TOP END Domain Gateways, gwgrp1 and gwgrp2, on two nodes. The gateways are not required to be on separate nodes; both may be configured on the same node.

```
*GROUPS
DMADMGRP LMID=mach1 GRPNO=1
gwgrp1 LMID=mach1 GRPNO=2
gwgrp2 LMID=mach2 GRPNO=3

*SERVERS
#GWTOPEND is the name of the TEDG binary program.
#Other parameters are strictly for illustrative purposes

DMADM      SRVGRP="DMADMGRP" SRVID=1001 REPLYQ=N RESTART=Y
           MAXGEN=5 GRACE=3600
GWADM      SRVGRP="gwgrp1" SRVID=1002 REPLYQ=N RESTART=Y MAXGEN=5
           GRACE=3600
GWTOPEND   SRVGRP="gwgrp1" SRVID=1003 RQADDR="gwgrp1" REPLYQ=N
           RESTART=Y MAXGEN=5 GRACE=3600
```

```
GWADM     SRVGRP="gwgrp2" SRVID=1004 REPLYQ=N RESTART=Y MAXGEN=5
          GRACE=3600
GWTOPEND  SRVGRP="gwgrp2" SRVID=1005 RQADDR="gwgrp2" REPLYQ=N
          RESTART=Y MAXGEN=5 GRACE=3600
```

See Also

- `UBBCONFIG(5)` in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*

Modifying the DMCONFIG File

Modify the `DMCONFIG` file to reflect the two gateway groups as shown in the following example.

```
*DM_LOCAL_DOMAINS
LDOM1 GWGRP="gwgrp1" DOMAINID="LDOM1" TYPE=TOPEND
LDOM2 GWGRP="gwgrp2" DOMAINID="LDOM2" TYPE=TOPEND

*DM_REMOTE_DOMAINS
RDOM1 DOMAINID="RDOM1" TYPE=TOPEND
RDOM2 DOMAINID="RDOM2" TYPE=TOPEND

*DM_TOPEND
LDOM1 NWADDR="//mach1:port1" TP_SYSTEM=SYSTEM1
LDOM2 NWADDR="//mach2:port2" TP_SYSTEM=SYSTEM2
RDOM1 NWADDR="//mach3:port3" TP_SYSTEM=SYSTEM1
RDOM2 NWADDR="//mach4:port4" TP_SYSTEM=SYSTEM2
```

This `GWTOPEND` Gateway . . .	**Communicates with This BEA TOP END System . . .**	**Represented by . . .**
`LDOM1`	`SYSTEM1`	`RDOM1`
`LDOM2`	`SYSTEM2`	`RDOM2`

See Also

- `DMCONFIG for GWTOPEND(5)` in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*

9 Sample Configuration File for a TOP END Domain Gateway

This topic includes the following sections:

- Sample Program Description
- Sample DMCONFIG File for the TEDG

See Also

- "Configuring the TOP END Domain Gateway" on page 4-1

Sample Program Description

This sample program, which is an extension of Example 1 in the core BEA Tuxedo `DMCONFIG(5)` reference page, shows five Bank Branch domains communicating with a Central Bank Branch. Three of the Bank Branches run within one BEA Tuxedo system domain (TDomain). The fourth Branch runs under the control of another TP domain and uses OSI TP to communicate with that domain. Example 1 has been extended to include a TEDG with a single connection to a BEA TOP END system. The

BEA TOP END system is running a banking application that offers services needed by the BEA Tuxedo application. Conversely, certain BEA Tuxedo services need to be available to BEA TOP END clients. A simple queuing example is also included.

Functions Available for the BEA TOP END EBANK Product

The BEA TOP END system, `BANKSYS`, offers a single product called `EBANK`. The following table lists the functions available for the `EBANK` product.

Table 9-1 Functions Available for BEA TOP END EBANK Product

EBANK Function	Description
`START`	Establish a connection with the bank application
`END`	Terminate the session
`LOGIN`	Log a user into the bank application
`LISTACCT`	List the user's accounts
`GETPAYES`	List electronic payment "payees"
`ELECPAY`	Perform an electronic payment
`BAL`	Obtain an account balance
`TRANSFER`	Transfer funds between accounts
`WITHDRAW`	Withdraw money from an account
`DEPOSIT`	Add money to an account
`REPORT`	Generate a report
`UPDATE`	Update the background

BEA Tuxedo Service Mappings for the EBANK Functions

The following DM_REMOTE_SERVICES mappings are used to make the functions shown in the table “Functions Available for BEA TOP END EBANK Product” available to the BEA Tuxedo application. Additionally, the BEA TOP END RTQ TEQNAME and two services accessed via the queue, REPORT and UPDATE, are made available to the BEA Tuxedo application.

Table 9-2 Mappings for EBANK Functions

Type	BEA Tuxedo Service	BEA TOP END Product and Function
SERVICE	te_start	EBANK, START
SERVICE	te_end	EBANK, END
SERVICE	te_login	EBANK, LOGIN
SERVICE	te_listacct	EBANK, LISTACCT
SERVICE	te_getpayees	EBANK, GETPAYES
SERVICE	te_elecpay	EBANK, ELECPAY
SERVICE	te_bal	EBANK, BAL
SERVICE	te_transfer	EBANK, TRANSFER
SERVICE	te_withdrawl	EBANK, WITHDRAW
SERVICE	te_deposit	EBANK, DEPOSIT
QSPACE	tuxqspace	TEQGROUP, TEQNAME
QNAME	te_report	EBANK, REPORT
QNAME	te_update	EBANK, UPDATE

SERVICE and QUEUE Mappings for the BEA Tuxedo Service “balance”

The following DM_LOCAL_SERVICES mappings are used to make the BEA Tuxedo service called balance available to the BEA TOP END system. Additionally, the BEA Tuxedo queue space qspace and its queue name qname are made available to the BEA TOP END system.

Table 9-3 Balance SERVICE and QUEUE Mappings

Type	BEA Tuxedo Service	BEA TOP END Product and Function
SERVICE	balance	TUX, BALANCE
QSPACE	qspace	TUXQUEUE, TUXQ
QNAME	qname	TUX, QSERV

Note: Because this sample configuration includes multiple gateway types, the LDOM parameter is specified in the DM_LOCAL_SERVICES section for the balance entry.

See Also

- “Configuration Guidelines for DM_LOCAL_SERVICES” on page 6-11
- DMCONFIG for GWTOPEND(5) in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*

Sample DMCONFIG File for the TEDG

The DMCONFIG file for the Central Bank Branch is shown below. Entries specific to the TEDG configuration are shown in **bold**.

```
# TUXEDO DOMAIN CONFIGURATION FILE FOR THE CENTRAL BANK
#
#
*DM_LOCAL_DOMAINS
# <local domain name> <Gateway Group name> <domain type> <domain
id> <log device>
#      [<audit log>] [<blocktime>]
#      [<log name>] [<log offset>] [<log size>]
#      [<maxrdtran>] [<maxtran>]
#      [<maxdatalen>] [<security>]
#      [<tuxconfig>] [<tuxoffset>]
#
#
DEFAULT: SECURITY = NONE


c01  GWGRP = bankg1
    TYPE = TDOMAIN
    DOMAINID = "BA.CENTRAL01"
    DMTLOGDEV = "/usr/apps/bank/DMTLOG"
    DMTLOGNAME = "DMTLG_C01"

c02  GWGRP = bankg2
    TYPE = OSITP
    DOMAINID = "BA.CENTRAL02"
    DMTLOGDEV = "/usr/apps/bank/DMTLOG"
    DMTLOGNAME = "DMTLG_C02"

c03 GWGRP = bankg3
    TYPE = TOPEND
    DOMAINID = "CENTRALBKGW"
    DMTLOGDEV = "/usr/apps/bank/DMTLOG"
    DMTLOGNAME = "DMTLG_C03"
    SECURITY = CLEAR

#
*DM_REMOTE_DOMAINS
#<remote domain name> <domain type> <domain id>
#
b01  TYPE = TDOMAIN
    DOMAINID = "BA.BANK01"

b02  TYPE = TDOMAIN
    DOMAINID = "BA.BANK02"

b03  TYPE = TDOMAIN
    DOMAINID = "BA.BANK03"

b04  TYPE = OSITP
```

```
    DOMAINID = "BA.BANK04"

b05 TYPE = TOPEND
    DOMAINID = "BANK05"

*DM_TDOMAIN
#
# <local or remote domain name> <network address> [<nwdevice>]
#
# Local network addresses
c01  NWADDR = "//newyork.acme.com:65432"  NWDEVICE ="/dev/tcp"

# Remote network addresses
b01  NWADDR = "//192.11.109.5:1025" NWDEVICE = "/dev/tcp"
b02  NWADDR = "//dallas.acme.com:65432" NWDEVICE = "/dev/tcp"
b03  NWADDR = "//192.11.109.156:4244" NWDEVICE = "/dev/tcp"

*DM_OSITP
#
#<local or remote domain name> <apt> <aeq>
#   [<aet>] [<acn>] [<apid>] [<aeid>]
#   [<profile>]
#
c02  APT = "BA.CENTRAL02"
   AEQ = "TUXEDO.R.4.2.1"
   AET = "{1.3.15.0.3},{1}"
   ACN = "XATMI"
b04  APT = "BA.BANK04"
   AEQ = "TUXEDO.R.4.2.1"
   AET = "{1.3.15.0.4},{1}"
   ACN = "XATMI"

*DM_TOPEND
#Local network addresses
c03  NWADDR = "//newyork.acme.com:65434"
     TP_SYSTEM = "BANKSYS"
#Remote network addresses
b05  NWADDR = "//sandiego.acme.com:65434"
     TP_SYSTEM = "BANKSYS"

*DM_LOCAL_SERVICES
#<service_name> [<Local Domain name>] [<access control>] [<exported
svcname>]
#       [<inbuftype>] [<outbuftype>]
#
#Not available to TOP END, no mapping
open_act ACL = branch LDOM=c01
close_act ACL = branch LDOM=c01
credit LDOM=c01
```

```
debit LDOM=c01
loan   LDOM = c02 ACL = loans

#Services exported to TOP END and other domains
balance TYPE=SERVICE TE_PRODUCT="TUX" TE_FUNCTION="BALANCE" LDOM =
c03

#Queues available to TOP END
qspace TYPE=QSPACE TE_RTQGROUP="TUXQUEUE" TE_RTQNAME="TUXQ"
LDOM=c03
qname TYPE=QNAME TE_PRODUCT="TUX" TE_FUNCTION="QSERV" LDOM=c03

*DM_REMOTE_SERVICES
#<service_name> [<Remote domain name>] [<local domain name>]
#        [<remote svcname>] [<routing>] [<conv>]
#        [<trantime>] [<inbuftype>] [<outbuftype>]
#
tlr_add LDOM = c01 ROUTING = ACCOUNT
tlr_bal LDOM = c01 ROUTING = ACCOUNT
tlr_add RDOM = b04 LDOM = c02 RNAME ="TPSU002"
tlr_bal RDOM = b04 LDOM = c02 RNAME ="TPSU003"

#
# New TOP END services available to TUXEDO
DEFAULT:      LDOM=c03 RDOM=b05
              TYPE=SERVICE TE_PRODUCT="EBANK"
te_start        TE_FUNCTION="START"
te_end          TE_FUNCTION="END"
te_login        TE_FUNCTION="LOGIN"
te_listacct     TE_FUNCTION="LISTACCT"
te_getpayees    TE_FUNCTION="GETPAYES"
te_elecpay      TE_FUNCTION="ELECPAY"
te_bal          TE_FUNCTION="BAL"
te_transfer     TE_FUNCTION="TRANSFER"
te_withdrawl    TE_FUNCTION="WITHDRAW"
te_deposit      TE_FUNCTION="DEPOSIT"

#
#TOP END RTQ queues available to Tuxedo
DEFAULT:      LDOM=c03 RDOM=b05 TYPE=QSPACE
tuxqspace TE_RTQGROUP="TEQGROUP" TE_RTQNAME="TEQNAME"

#
#TOP END services available to TUXEDO via tpenqueue and RTQ.
DEFAULT:      LDOM=c03 RDOM=b05 TYPE=QNAME
te_report     TE_PRODUCT="EBANK" TE_FUNCTION="REPORT"
te_update     TE_PRODUCT="EBANK" TE_FUNCTION="UPDATE"

*DM_ROUTING
```

```
# <routing criteria> <field> <typed buffer> <ranges>
#
ACCOUNT FIELD = branchid BUFTYPE ="VIEW:account"
       RANGES ="MIN - 1000:b01, 1001-3000:b02, *:b03"

*DM_ACCESS_CONTROL
#<acl name> <Remote domain list>
#
branch ACLIST = b01, b02, b03
loans  ACLIST = b04
```

See Also

- `DMCONFIG for GWTOPEND(5)`in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*

CHAPTER

10 Administering the TEDG During Run Time

This topic includes the following sections:

- Understanding Run-time Administration
- Using the dmadmin Command Interpreter

Understanding Run-time Administration

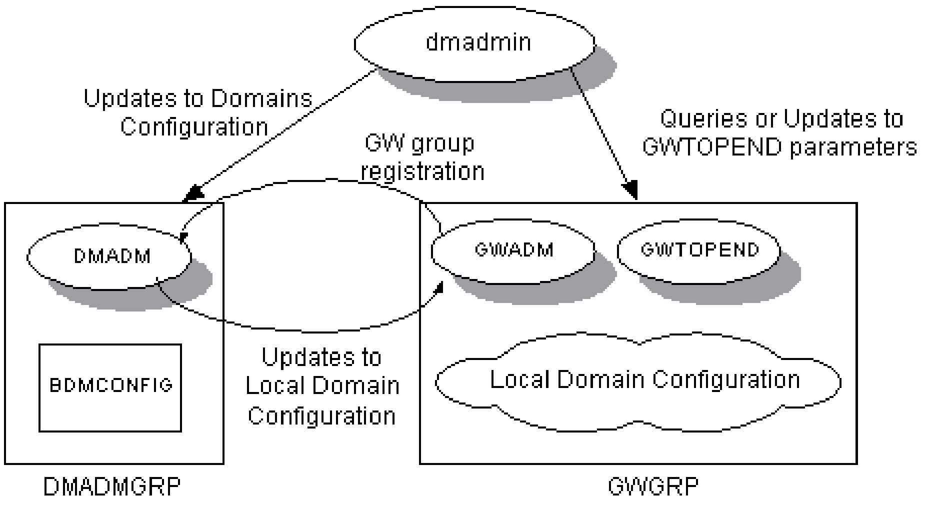
The main tools for run-time administration are:

- A command, `dmadmin(1)`, that allows administrators to configure, monitor, and tune domain gateway groups dynamically.
- A Domains administrative server, `DMADM(5)`, that provides the administrative processing required for updating a Domains configuration. This server acts as a back-end to the `dmadmin` command.
- A gateway administrative server, `GWADM(5)`, that provides the run-time administrative processing required for a specific gateway group. This server also acts as a back-end to the `dmadmin` command.

Note: The `GWTOPEND(5)` gateway process provides connectivity to remote gateway processes. Clients and servers send and receive messages between BEA Tuxedo and BEA TOP END systems via the `GWTOPEND` process. This process is not involved in Domains administration.

The following illustration shows how these tools are used in run-time administration.

Figure 10-1 Domains Run-time Administration


dmadmin
Updates to Domains Configuration
GW group registration
Queries or Updates to GWTOPEND parameters
DMADM
BDMCONFIG
Updates to Local Domain Configuration
GWADM
GWTOPEND
Local Domain Configuration
DMADMGRP
GWGRP

See Also

- `dmadmin(1)` in *BEA Tuxedo Command Reference*
- `DMADM(5)` in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*
- `GWADM(5)` in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*
- `GWTOPEND(5)` in the *File Formats, Data Descriptions, MIBs, and System Processes Reference*

Using the dmadmin Command Interpreter

The `dmadmin` command is an interactive command interpreter for the administration of domain gateway groups defined for a particular BEA Tuxedo application. It is used by application administrators for the interactive administration of the information stored in the `BDMCONFIG` file and the different gateway groups running within a particular BEA Tuxedo application.

Use `dmadmin` to:

- Obtain statistics or other information gathered by gateway groups
- Change the gateway group parameters
- Add (or update) information to the `BDMCONFIG` file

To use `dmadmin` in configuration mode, execute it in either of two ways:

- Use the `-c` option as follows.

  ```
  dmadmin -c
  ```

- Use the `config` subcommand as follows.

  ```
  dmadmin
  >config
  ```

Defining a BEA TOP END Password

For the TOP END Domain Gateway, if security is enabled, you need to associate a BEA TOP END password with each local domain of type `TOPEND` defined in the `DM_LOCAL_DOMAINS` section of the `DMCONFIG` file. You assign this password using the `dmadmin topendpasswd` subcommand as follows.

`topendpasswd (tepasswd) [-r]` *`local_domain_name`*

This command prompts the administrator for a new password for the specified local domain of type `TOPEND`. The gateway uses this password when sending messages to the BEA TOP END system. The `-r` option specifies that existing passwords and new passwords should be encrypted using a new key generated by the system. The password is truncated if it is longer than the maximum length of twelve characters.

For details, see `dmadmin(1)` in the *BEA Tuxedo Command Reference*.

Run-time Deletions

Run-time deletions to the `BDMCONFIG` file can be performed only when the changes do not involve an active gateway group.

See Also

- `dmadmin(1)` in the *BEA Tuxedo Command Reference*
- `topendpasswd(1)` in the *BEA Tuxedo Command Reference*

Part III Programming Considerations

CHAPTER

11 API Programming

This topic includes the following sections:

- Using the BEA Tuxedo ATMI (API) with the TEDG
- Using the BEA TOP END CSI with the TEDG

See Also

- "Characteristics of the APIs" on page 2-13

Using the BEA Tuxedo ATMI (API) with the TEDG

The BEA Tuxedo application programming interface (API) that you use to communicate with BEA TOP END is called the Application-to-Transaction Monitor Interface, or ATMI. BEA Tuxedo client and server programs use ATMI functions to communicate with BEA TOP END client and server programs through the TOP END Domain Gateway (TEDG).

To a great extent, the way in which ATMI functions are used to communicate with BEA TOP END applications is identical to how ATMI functions are used to communicate with other BEA Tuxedo programs or applications through other domain gateways. The primary difference is that the TEDG does not support the same functions and features as other domain gateways. Because the TEDG connects two similar but not identical client/server environments, only features common to both environments can be supported when these environments are interoperating. This set of features common to both the BEA Tuxedo and BEA TOP END systems is provided by the TEDG through configuration, feature mapping, and exposing supported features at the ATMI programming interface in a way that closely matches the standard use of the ATMI.

As with other domain gateways, the TEDG is viewed by the BEA Tuxedo client as a BEA Tuxedo server; it maps requests from BEA Tuxedo client programs to BEA TOP END servers. When a BEA TOP END client makes a request, the TEDG maps the request and sends it to a BEA Tuxedo server. For this reason, the TEDG is viewed as a BEA Tuxedo client by the BEA Tuxedo server.

From a programming perspective, the TEDG is transparent to BEA Tuxedo ATMI applications; it looks like a BEA Tuxedo application and is accessed through the same ATMI functions used to access any BEA Tuxedo application. As with all applications that communicate, a BEA Tuxedo application and a BEA TOP END application must be designed to work together on requests, responses, and error handling. Because the TEDG configuration and mapping affect this interaction, the application designer must understand the TEDG configuration.

"Characteristics of the BEA Tuxedo ATMI" on page 2-13 lists the supported and unsupported features of the BEA Tuxedo ATMI.

Limitations of Supported ATMI Features

Some of the supported ATMI features have limitations and special requirements when used for communication via the TOP END Domain Gateway. These limitations and requirements ensure that BEA Tuxedo and BEA TOP END applications have a common method of communicating. Some of the limitations are:

- Conversational communication is limited to the following format:
  - One process sends a single message and then releases control of the conversation.
  - Another process responds with a single message and then releases control of the conversation.
- Because the features provided by BEA TOP END RTQ and BEA Tuxedo queuing are not identical, several options in each API are unavailable.

- The BEA Tuxedo system supports embedded FML32 buffers; the BEA TOP END system does not. These buffers are ignored by the TEDG and allowed to go through the gateway. If embedded FML32 buffers are used in a BEA Tuxedo application that is accessed by BEA TOP END clients, however, a failure may occur on the BEA TOP END node.
- Public/private key encryption of application messages is supported between the BEA Tuxedo application and the TEDG but does not apply to messages between the TEDG and a BEA TOP END application. BEA Tuxedo configuration options that require the use of encryption are enforced. The ability to auto-encrypt messages from a BEA TOP END application is provided. Public/private key encryption for the TEDG is handled the same way that this type of encryption is handled for other non-TDomains BEA Tuxedo gateways.
- Digital signatures are supported for application messages sent between the BEA Tuxedo application and the TEDG but are not supported for messages between the TEDG and a BEA TOP END application. BEA Tuxedo configuration options that require use of encryption are enforced. The ability to auto-sign messages from a BEA TOP END application is provided. Digital signatures for the TEDG are handled the same way that digital signatures are handled for other non-TDomains BEA Tuxedo gateways.

See Also

- "Characteristics of the APIs" on page 2-13
- "Introduction to the C Language Application-to-Transaction Monitor Interface" on page -7 in the *BEA Tuxedo ATMI C Function Reference*

Using the BEA TOP END CSI with the TEDG

The BEA TOP END application programming interface (API) is called the Client/Server Interaction facility, or CSI. BEA TOP END client and server programs use CSI routine calls to communicate with BEA Tuxedo client and server programs through the TOP END Domain Gateway (TEDG). To a great extent, the way in which CSI routine calls are used to communicate with BEA Tuxedo ATMI applications is identical to how CSI routines are used to communicate with other CSI applications. The primary difference is the features or options supported for each routine. The TEDG supports a set of features common to both the BEA Tuxedo and BEA TOP END systems through configuration, feature mapping, and exposing supported features at the CSI programming interface in a way that closely matches the standard use of the CSI.

The TEDG is viewed by the BEA TOP END client as a BEA TOP END server and maps requests from BEA TOP END client programs to a BEA Tuxedo server. When a BEA Tuxedo client makes a request, the TEDG maps the request and sends it to a BEA TOP END server. For this reason, the TEDG is viewed as a BEA TOP END client by a BEA TOP END server.

From a programming perspective, the TEDG is transparent to BEA TOP END applications: it functions like a BEA TOP END application and is accessed using the same CSI calls used to access a BEA TOP END application. As with all applications that communicate, both the BEA TOP END application and the BEA Tuxedo application must be designed to work together on requests, responses, and error handling. Because the TEDG configuration and mapping affect this interaction, the application designer must understand the TEDG configuration.

For lists of supported and unsupported features of the BEA TOP END CSI, see "Characteristics of the BEA TOP END CSI" on page 2-15.

Limitations of Supported CSI Facilities

Some of the supported CSI features and interfaces have limitations and special requirements when used for communicating via the TOP END Domain Gateway. These limitations and requirements ensure that BEA Tuxedo and BEA TOP END applications have a common method of communicating. Some of the limitations are:

- Conversational communication with an application context is accomplished by mapping to the BEA Tuxedo conversational paradigm. There are limitations to the use of function qualifiers with the TEDG. A traditional BEA TOP END application may use function qualifiers to indicate the step of the conversational interaction; this is not possible with the TEDG. Successful communication in a conversational manner requires adapting one or both of the applications to the features available through the TEDG mapping of context to conversations.

- Because the features provided by BEA TOP END RTQ and BEA Tuxedo queuing are not identical, a number of options in each API are unavailable.

See Also

- `tp_intro(3T)` in the *BEA TOP END Programmer's Reference Manual*

CHAPTER

12 Request/Response Mode Programming

This topic includes the following sections:

- Using Request/Response Messaging with the TEDG
- How Messages Are Passed from BEA Tuxedo Clients to BEA TOP END Servers
- How Messages Are Passed from BEA TOP END Clients to BEA Tuxedo Servers

Using Request/Response Messaging with the TEDG

The TEDG supports the exchange of messages in request/response mode between BEA Tuxedo and BEA TOP END systems. Messages are passed between the two systems using the TEDG as an intermediary.

See Also

- "Request/Response Message Passing" on page 2-3

How Messages Are Passed from BEA Tuxedo Clients to BEA TOP END Servers

BEA Tuxedo clients communicate with BEA TOP END services through the TEDG. Relative to a BEA Tuxedo client, the TEDG functions as a local server. The TEDG advertises services specified in the `SERVICE` entries in the `DM_REMOTE_SERVICES` section of the `DMCONFIG` file. Client programs make requests and receive responses using `tpcall(3c)` in the same way they use `tpcall()` to interact with local services. Asynchronous requests and responses are supported through the `tpacall(3c)` and `tpgetrply(3c)` functions. If, in the `DM_REMOTE_SERVICES` section of the `DMCONFIG` file, the `SERVICE` entry for a TEDG service contains the setting `CONV=N`, then BEA Tuxedo clients may communicate with that service in only one mode: request/response. (`CONV=N` is the default value and does not need to be set.)

The TEDG uses the requested service name to locate the `SERVICE` entry in the `DM_REMOTE_SERVICES` section to determine the corresponding BEA TOP END product, function, Message Sensitive Routing (MSR) target, and function qualifier for the mapped BEA TOP END request. Data marshalling is performed by the TEDG, as required, before the message is sent to the BEA TOP END system. The BEA TOP END system then routes the request to a server on a BEA TOP END node that has received the message. The BEA TOP END server response is unmarshalled, if necessary. The TEDG maps the status of the request and prepares the buffer for delivering the response to the BEA Tuxedo client.

How the TEDG Works with BEA Tuxedo Clients

Client operations are programmed with the same functions used for any BEA Tuxedo client.

Use this function . . .	**For . . .**
`tpcall()`	Synchronous requests and responses
`tpacall()`	Asynchronous requests and responses

Use this function . . .	For . . .
`tpgetreply()`	Asynchronous requests and responses

These functions are programmed in the normal manner.

As a BEA Tuxedo client programmer, you need to know the following information:

- The `service` name assigned by the administrator (in the `DMCONFIG` file) to the BEA TOP END service.
- The buffer type required by the BEA TOP END service. A BEA Tuxedo buffer type of `CARRAY` or `X_OCTET` is used to prepare a raw buffer required by the BEA TOP END service. Buffers of these types have no data marshalling support when transmitted by the TEDG between BEA TOP END nodes of different types.

  If a BEA TOP END service supports `FML32` input, the BEA Tuxedo client uses the BEA Tuxedo `FML32` buffer type. `FML32` is beneficial because the TEDG and the BEA TOP END system support data marshalling of these buffers when the buffers are transmitted between the TEDG and a BEA TOP END node of a different type.

  A BEA TOP END service may support one or more of these buffer types.
- Whether the administrator has required, in the `DMCONFIG` file, the exclusive use of one buffer type. If a buffer of a type that is inappropriate (due to the fact that it does not match either the configured type or any type supported by the TEDG) is sent to the TEDG, a call to `tpcall()` or `tpgetrply()` returns a `TPESVCERR` error (not a `TPEITYPE` error).

Because BEA TOP END messages are limited to 30K bytes, the size of the client request may not exceed that limit. For an `FML32` message, the limit applies after the FML index is stripped from the message.

How the TEDG Maps Client Requests

A client request may be transactional or non-transactional and it may or may not require a response. The following table shows how BEA Tuxedo client flags are mapped to a BEA TOP END request. All other flags (`TPNOCHANGE`, `TPNOBLOCK`, `TPNOTIME`, `TPSIGRSTRT`) and the service priority (see `tpsprio()`) either are local to the application or affect only client-TEDG interactions in the BEA Tuxedo system. By processing the following flags, the TEDG accomplishes tasks that are normally done in the BEA Tuxedo system.

Table 12-1 BEA Tuxedo Client Flag Mapping

BEA Tuxedo Client Flag	Action
`TPNOTRAN`	The TEDG preserves the fact that the request is excluded from the client transaction by the ATMI library.
`TPNOREPLY`	Passed as `TP_NO_RESPONSE` to the BEA TOP END system. The BEA TOP END routing status is not passed to the BEA Tuxedo client.

The `tperrno` values returned to the BEA Tuxedo client are standard values. Because the TEDG acts as a BEA Tuxedo server, it also returns `TPESVCERR` and `TPESVCFAIL` for certain conditions. The BEA Tuxedo client cannot interpret these messages as "application defined errors" exclusively:

- `TPESVCERR` indicates delivery errors and other general errors.
- `TPESVCFAIL` indicates that the BEA TOP END system/service reset the mapped dialog request. Reset is one method used by a BEA TOP END server to indicate that it received a request but cannot process it.

For additional information, see "Error Values" on page 12-6.

Note: `tpurcode` is not supported by the TEDG.

The TEDG maps the response from the BEA TOP END system/server to a response that the BEA Tuxedo client accesses through the `tpcall` parameters or the `tpgetrply` function. A BEA TOP END server may deliver a response in either a raw buffer or an `FML32` buffer. A raw buffer is normally mapped by the TEDG to a `CARRAY` buffer unless the administrator configures it to map to an `X_OCTET` buffer. Additionally, the administrator may constrain the response buffer to one of the following types: `CARRAY`, `X_OCTET` or `FML32`. If the BEA TOP END service returns an incompatible buffer, the TEDG returns a `tperrno` of `TPEOTYPE.`

How the TEDG Works with BEA TOP END Servers

Relative to a BEA TOP END server, the TEDG functions as a BEA TOP END client: it receives mapped BEA Tuxedo client requests in the normal manner through `tp_server_receive(3T)`. The buffer received is either a raw buffer or an `FML32` buffer, depending on the message sent by the BEA Tuxedo client. The BEA TOP END server receives one of the following types of requests:

- A normal request.
- A "no-response" request—this type of request (in which the `TP_NO_RESPONSE` flag is set) is received when the BEA Tuxedo client sends a `TPNOREPLY` message.

The BEA TOP END server handles both types of requests according to standard BEA TOP END programming requirements. The client request may be transactional or non-transactional.

How the TEDG Maps BEA TOP END Server Send Flags

The BEA TOP END server responds to a client request using `tp_server_send(3T)`. The response buffer may be either a raw buffer or an `FML32` buffer, depending on what is supported by the BEA TOP END application, the TEDG configuration, and the BEA Tuxedo client. This buffer is mapped to a BEA Tuxedo client buffer as described in "How the TEDG Maps Client Requests" on page 12-3.

The following table shows how BEA TOP END `tp_server_send(3T)` flags are mapped. The server may indicate an error by resetting the dialog or by responding with an application-defined field value in the response buffer. The BEA Tuxedo client must be programmed to respond appropriately. Do not use the `output_format` and `attach_info` parameters on responses to the TEDG; they are not supported.

Table 12-2 BEA TOP END "Server Send" Flags

BEA TOP END Server Flag	Action
`TP_RESET_DIALOG`	`TPESVCFAIL` is returned to the BEA Tuxedo client.
`TP_ROLLBACK_ONLY`	The TEDG marks the state of the transaction associated with the request as "abort-only."

Table 12-2 BEA TOP END "Server Send" Flags (Continued)

BEA TOP END Server Flag	Action
`TP_DISSOLVE`	A response is passed to the BEA Tuxedo client. Because the TEDG manages dialogs in a special manner, the results of setting the flag are the same as the results of not setting the flag.
`TP_APPL_CONTEXT`	Indicates that the server wants to initiate a conversation. This request for conversational mode is an error for request/response processing. `TESVCERR` is returned to the BEA Tuxedo client. The BEA TOP END server is disconnected (`TP_DISCONNECT`) by the TEDG.
`TP_FML_BUF`	The TEDG passes the user data as an `FML32` buffer.

Error Values

The following error values may be returned to a BEA Tuxedo client because problems exist in the TEDG, the BEA TOP END system, or the BEA TOP END server. Keep in mind that a single `tperrno` value may be used to report any one of many possible causes of the error being reported.

Because the TEDG does not advertise services based upon the actual availability of BEA TOP END services, a message may be routed to a BEA TOP END node where the services actually are unavailable, resulting in a `tperrno` of `TPENOENT`, while other routing decisions may result in successful requests. If a service is to be available on multiple nodes, the design of the BEA Tuxedo application, the BEA TOP END application, and the TEDG must take into account the possibility that this type of failure may occur. A well-designed application, that ensures that there are multiple, restartable copies of the servers, reduces the possibility of such errors occurring.

Table 12-3 Error Values Returned to a BEA Tuxedo Client

BEA Tuxedo tperrno Value	Cause
`TPENOENT`	No match was found on the `SERVICE` entry.
`TPENOENT`	A `tpcall` or `tpacall` request was sent to a conversational service (`CONV=Y`).

Table 12-3 Error Values Returned to a BEA Tuxedo Client (Continued)

BEA Tuxedo tperrno Value	Cause
TPENOENT	TP_SERVICE was returned by the BEA TOP END system.
TPESVCERR with an error detail of TPED_DOMAINUNREACHABLE	An on-demand connection failed.
TPESVCERR	The input buffer type does not match the one specified in the INBUFTYPE field of the SERVICE entry or it is not a type that is supported by the TEDG. (FML32, CARRAY, and X_OCTET are supported.)
TPESVCERR	The input buffer exceeds the maximum message size of 30K bytes (after the FML index is stripped for FML buffers).
TPESVCERR	The server returned TP_APPL_CONTEXT on a non-conversational SERVICE entry.
TPESVCFAIL	TP_RESET was returned by the BEA TOP END system.
TPEOTYPE	The buffer type of the server response does not match the type specified in the OUTBUFTYPE field of the SERVICE entry.
TPESYSTEM	Error due to public/private key encryption: The client input was rejected because the BEA Tuxedo system is configured to require encryption and the TEDG could not decrypt the message before sending it to the BEA TOP END system.
TPESYSTEM	Error due to digital signature: The client input was rejected because the BEA Tuxedo system is configured to require digital signatures and the TEDG could not remove the digital signature from the message before sending the message to the BEA TOP END system.

See Also

- `tpacall(3c)` in the *BEA Tuxedo ATMI C Function Reference*
- `tpcall(3c)` in the *BEA Tuxedo ATMI C Function Reference*
- `tpgetrply(3c)` in the *BEA Tuxedo ATMI C Function Reference*
- `tp_server_receive(3T)` in the *BEA TOP END Programmer's Reference Manual*
- `tp_server_send(3T)` in the *BEA TOP END Programmer's Reference Manual*

How Messages Are Passed from BEA TOP END Clients to BEA Tuxedo Servers

BEA TOP END clients communicate with BEA Tuxedo servers through the TEDG. If, in the `DM_LOCAL_SERVICES` section of the `DMCONFIG` file, the `SERVICE` entry for a TEDG service contains the setting `CONV=N`, then BEA TOP END clients may communicate with that service in only one mode: request/response. (`CONV=N` is the default value and does not need to be set.)

Relative to a BEA TOP END client, the TEDG functions as a BEA TOP END server on a remote BEA TOP END node. The TEDG receives a BEA TOP END client request and maps it to a corresponding BEA Tuxedo request. Note that the BEA Tuxedo system may route the request to a BEA Tuxedo server anywhere within the BEA Tuxedo configuration.

The process for passing messages in request/response mode from a BEA TOP END client to a BEA Tuxedo server is as follows:

- When a network connection is made, the TEDG advertises services listed in the `SERVICE` entries in the `DM_LOCAL_SERVICES` section of the `DMCONFIG` file. The names of the advertised product, function, and MSR target are used in normal BEA TOP END routing and load-balancing algorithms to determine the destination node.

- The message includes the following information: the names of the product, function, optional MSR target, and optional function qualifier. When the TEDG receives the message, it uses this information to search the `SERVICE` entries in the `DMCONFIG` file for the appropriate BEA Tuxedo service.
- The TEDG performs any required unmarshalling of data.
- The message is sent to the appropriate BEA Tuxedo service.
- The BEA Tuxedo server response is mapped to a BEA TOP END status and buffer, marshalled (if necessary), and sent to the BEA TOP END client.

How the TEDG Works with BEA TOP END Clients

Client operations are programmed with the same functions used for any BEA TOP END client.

Use This API . . .	For . . .
`tp_client_send`	Making asynchronous requests
`tp_client_signon`	Making asynchronous requests
`tp_client_receive`	Receiving responses

These functions are used in the normal manner when making a service request to a BEA Tuxedo server through the TEDG.

As a BEA TOP END client programmer, you need to know the following information:

- The product and function name assigned by the administrator, in the `DMCONFIG` file, to the BEA Tuxedo service.
- If you are trying to match existing BEA TOP END programming, the function qualifier value, optionally assigned by the administrator.

- The MSR target name, which may have been assigned by the administrator in the DMCONFIG file. (Assigning an MSR target is optional.) This MSR target name may be used:
  - To match a configured BEA TOP END MSR routing strategy, or
  - Directly by the BEA TOP END client application.
- The buffer type required by the BEA Tuxedo service. A raw message must be sent if the BEA Tuxedo server requires either CARRAY or X_OCTET. By default a raw message is mapped to a CARRAY buffer. Buffers of these types have no data marshalling support when transmitted to the TEDG by a BEA TOP END node of a different type.

  If a BEA Tuxedo service supports FML32 input, the BEA TOP END client must use the FML32 message type. FML32 is beneficial because the TEDG and the BEA TOP END system support data marshalling of these buffers when the buffers are transmitted between the TEDG and a BEA TOP END node of a different type.

  A BEA Tuxedo service may support one or more of these buffer types.
- Whether the administrator has required, in the DMCONFIG file, the exclusive use of one buffer type. If a buffer of a type that is inappropriate (due to the fact that it does not match either the configured type or any type supported by the TEDG) is sent to the TEDG, a tp_client_receive call returns a TP_RESET error.

How the TEDG Maps Client Requests

A client request may be transactional or non-transactional and it may or may not require a response. The following table shows how BEA TOP END client flags are mapped. By processing these flags, the TEDG accomplishes tasks that are normally done in the BEA TOP END system. Do not use the input_format and attach_info parameters on requests to the TEDG; they are not supported.

Table 12-4 BEA TOP END Client Flag Mapping

BEA TOP END Client Flag	Action
TP_DISSOLVE	The dialog is dissolved after the TEDG returns the BEA Tuxedo server response.
TP_NON_TRANSACT	The TEDG preserves the fact that the request is excluded from the client transaction by the CSI library.

Table 12-4 BEA TOP END Client Flag Mapping (Continued)

BEA TOP END Client Flag	Action
`TP_NO_RESPONSE`	Mapped to `TPNOREPLY` by the TEDG. Like the BEA TOP END system, the TEDG responds to the BEA TOP END client with a routing status.
`TP_FML_BUF`	The TEDG passes the user data as an `FML32` buffer.

The status and extended status values returned to the BEA TOP END client are standard values. Additional information on the mapping of error values is provided later in this topic.

The TEDG maps the response from the BEA Tuxedo system/server to a response that the BEA TOP END client accesses through the `tp_client_receive(3T)` call. A BEA Tuxedo server may deliver a response in a `CARRAY`, `X_OCTET`, or `FML32` buffer. A `CARRAY` or `X_OCTET` buffer is mapped by the TEDG to a raw message; an `FML32` buffer, to a BEA TOP END `FML32` message. Additionally, the administrator may constrain the response buffer to one of the following types: `CARRAY`, `X_OCTET`, or `FML32`. If the BEA Tuxedo service returns an incompatible buffer, the TEDG returns a `TP_RESET` status.

How the TEDG Works with BEA Tuxedo Servers

Relative to a BEA Tuxedo server, the TEDG functions as a BEA Tuxedo client: it receives mapped BEA TOP END client requests in the normal manner. The type of the buffer it receives (`CARRAY`, `X_OCTET`, or `FML32`) depends on the message sent by the client. The BEA Tuxedo server processes the request in the normal manner. The request may be transactional or non-transactional. It may be a "no-reply" request.

A BEA Tuxedo server responds to a client request using `tpreturn(3c)`. `TPSUCCESS` is mapped to a `TP_OK` BEA TOP END status. `TPFAIL` and `TPEXIT`, along with a number of other error conditions, are mapped to a `TP_RESET` BEA TOP END status. A reply message is supported only with `TPSUCCESS`.

Note: The application-defined return code, *`rcode`*, is not supported by the TEDG.

The BEA Tuxedo server may deliver a reply in a CARRAY, X_OCTET, or FML32 buffer, depending on which buffer types are supported by the BEA TOP END system, the TEDG configuration, and the BEA TOP END client. The reply buffer is mapped to a BEA TOP END message. The server may indicate an error by sending a TPFAIL error or by responding with an application-defined field value in the reply buffer. The BEA TOP END client must be programmed to handle that error reporting interface of the server.

Because BEA TOP END messages are limited to 30K bytes, the BEA Tuxedo server reply may not exceed that limit. For an FML32 message, the limit applies after the FML index is stripped from the message.

Error Values

In addition to regular BEA TOP END error status messages, a number of other status messages may be returned to a BEA TOP END client as a result of problems in the TEDG, the BEA Tuxedo system, or the BEA Tuxedo server. Keep in mind that a single error status value may be used to report any one of many possible causes of the error being reported.

Because the TEDG does not advertise services based upon the actual availability of BEA Tuxedo services, a message may be routed to a BEA Tuxedo node where the services actually are unavailable, resulting in a TP_SERVICE error, while other routing decisions may result in successful requests. If a service is to be available on multiple nodes, the design of the BEA Tuxedo application, the BEA TOP END application, and the TEDG must take into account the possibility that this type of failure may occur. A well-designed application, that ensures that there are multiple, restartable copies of the servers, reduces the possibility of such errors occurring.

Table 12-5 Error Values Returned to a BEA TOP END Client

BEA TOP END Error Status	Cause
TP_SERVICE, TP_EXT_MSR_FAILURE	The TEDG SERVICE entry lookup failed: the target was not found.
TP_SERVICE, TP_EXT_NO_SUCH_SERV	The TEDG SERVICE entry lookup failed: the product or function was not found (ignoring entries for which RDOM does not have access).

Table 12-5 Error Values Returned to a BEA TOP END Client (Continued)

BEA TOP END Error Status	Cause
TP_SERVICE, TP_EXT_NO_SUCH_SERV	TPENOENT was returned by the BEA Tuxedo system.
TP_RESET, TP_EXT_SERVER_APPL	The input buffer type does not match the type specified in the INBUFTYPE field of the SERVICE entry.
TP_RESET, TP_EXT_SERVER_APPL	The server response buffer type does not match the type specified in the OUTBUFTYPE field of the SERVICE entry or it is not a type supported by the TEDG. (The supported types are FML32, CARRAY, and X_OCTET.)
TP_RESET, TP_EXT_SERVER_APPL	The server response buffer exceeds the maximum message size of 30K bytes (after the FML index is stripped for FML buffers).
TP_RESET, TP_EXT_SERVER_APPL	TPESVCERR, TPESVCFAIL, and all other tperrno values.
TP_RESET, TP_EXT_SERVER_APPL	Error due to public/private key encryption: The client input was rejected by the BEA Tuxedo server because the BEA Tuxedo system is configured to require encryption and the TEDG could not decrypt the message before sending it to the BEA TOP END system.
TP_RESET, TP_EXT_SERVER_APPL	Error due to digital signature: The client input was rejected by the BEA Tuxedo server because the BEA Tuxedo system is configured to require digital signatures and the TEDG could not remove the digital signature from the message before sending the message to the BEA TOP END system.

See Also

- `tpreturn(3c)` in the *BEA Tuxedo ATMI C Function Reference*

- `tp_client_receive(3T)` in the *BEA TOP END Programmer's Reference Manual*
- `tp_client_send(3T)` in the *BEA TOP END Programmer's Reference Manual*
- `tp_client_signon(3T)` in the *BEA TOP END Programmer's Reference Manual*

13 Conversational Mode Programming

This topic includes the following sections:

- Using Conversational Messaging with the TEDG
- How Messages Are Passed from BEA Tuxedo Clients to BEA TOP END Servers
- How Messages Are Passed from BEA TOP END Clients to BEA Tuxedo Servers

Using Conversational Messaging with the TEDG

The BEA Tuxedo and BEA TOP END systems support different styles of conversational message passing:

- The BEA TOP END system supports pseudo-conversations. A BEA TOP END server can indicate that it wants to maintain context (a conversation) with a client at almost any time by setting the `TP_APPL_CONTEXT` flag. (See `tp_server_send(3T)` in the *BEA TOP END Programmer's Reference Manual* for details.)
- BEA Tuxedo conversations are much more structured. The administrator configures servers to be conversational and a set of conversational functions

used by the client and server establishes a connection and maintains the conversation. BEA Tuxedo conversations are half-duplex meaning that one side of the conversation can send multiple messages before relinquishing control of the connection.

The TEDG provides a mapping of BEA TOP END pseudo-conversations to the BEA Tuxedo conversation model. This mapping enables BEA Tuxedo clients to maintain context with BEA TOP END servers for multiple-step interactions. The reverse is also true for BEA TOP END clients that want to maintain context with BEA Tuxedo servers. Multiple-step interactions are routed to the same server until the conversation is terminated. Single-step interactions between conversational components are also allowed. If you need the same server to handle single-step and multiple-step interactions, then you must use conversational messaging.

See Also

- `tp_server_send`(3T) in the *BEA TOP END Programmer's Reference Manual*

How Messages Are Passed from BEA Tuxedo Clients to BEA TOP END Servers

BEA Tuxedo clients communicate conversationally with BEA TOP END services through the TEDG. Relative to a BEA Tuxedo client, the TEDG functions as a local conversational server. The TEDG advertises services specified in the `SERVICE` entries in the `DM_REMOTE_SERVICES` section of the `DMCONFIG` file. If a BEA TOP END server supports pseudo-conversations, then the BEA Tuxedo administrator must configure the services offered by that server as conversational by setting `CONV=Y` in the `SERVICE` entry in the `DM_REMOTE_SERVICES` section of the `DMCONFIG` file. If the `SERVICE` entry for a TEDG service is `CONV=Y`, then BEA Tuxedo clients may communicate with that service only in conversational mode. The BEA Tuxedo client uses the `tpconnect(3c)`, `tpsend(3c)`, `tprecv(3c)`, and `tpdiscon(3c)` functions.

The TEDG uses the requested service name to locate the `SERVICE` entry in the `DM_REMOTE_SERVICES` section to determine the corresponding BEA TOP END product, function, MSR target and function qualifier for the mapped BEA TOP END request. Data marshalling is performed by the TEDG, as required, before the message is sent to the BEA TOP END system. The BEA TOP END system then routes the request to a server on a BEA TOP END node that has received the message. The BEA TOP END server response is unmarshalled, if necessary. The TEDG maps the status of the request and prepares the buffer for delivering the response to the BEA Tuxedo client.

The BEA TOP END server may use application context, on an ongoing basis, to determine whether to continue a conversation with a BEA Tuxedo client: as long as application context is present, the conversation is maintained; when the application context is absent, the conversation is ended.

How the TEDG Works with BEA Tuxedo Clients

The BEA Tuxedo client programmer uses `tpconnect(3c)` to establish a conversation with a BEA TOP END server. The server may respond conversationally or it may respond with a single response and end the conversation. Unlike clients participating in normal BEA Tuxedo conversations, clients accessing the BEA TOP END system must accept a response to each message. To prepare a client to receive a response to each request, set the `TPRECVONLY` flag on `tpconnect()` and subsequent `tpsend(3c)` calls. When this flag is set, the client waits to receive a message before sending another. The BEA Tuxedo `tprecv(3c)` function is used to receive replies from the BEA TOP END system. If `TPRECVONLY` is not specified or if the flags are set to `TPSENDONLY`, the TEDG rejects the request, logs an error, and returns a `TPEV_SVCERR` event to the client. The `tpconnect()` and `tpsend()` calls are mapped to the equivalent of a BEA TOP END `tp_client_send(3T)` call, regardless of whether or not there is any data. If there is no data, the server receives a zero-length message.

As a BEA Tuxedo client programmer, you need to know the following information:

- The service name assigned by the administrator (in the `DMCONFIG` file) to the BEA TOP END service.
- The buffer type required by the BEA TOP END service. A BEA Tuxedo buffer type of `CARRAY` or `X_OCTET` is used to prepare a raw buffer required by the BEA TOP END service. Buffers of these types have no data marshalling support when transmitted by the TEDG between BEA TOP END nodes of different types.

If a BEA TOP END service supports `FML32` input, the BEA Tuxedo client uses the BEA Tuxedo `FML32` buffer type. `FML32` is beneficial because the TEDG and the BEA TOP END system support data marshalling of these buffers when the buffers are transmitted between the TEDG and a BEA TOP END node of a different type.

A BEA TOP END service may support one or more of these buffer types.

- Whether the administrator has required, in the `DMCONFIG` file, the exclusive use of one buffer type. If a buffer of a type that is inappropriate (due to the fact that it does not match either the configured type or any type supported by the TEDG) is sent to the TEDG, a call to `tprecv()` returns a `TPEV_SVCERR` event.

Because BEA TOP END messages are limited to 30K bytes, the size of the client request may not exceed that limit. For an `FML32` message, the limit applies after the FML index is stripped from the message.

How the TEDG Maps Clients Requests

A client request may be transactional or non-transactional. The following table shows how BEA Tuxedo client flags are mapped to a BEA TOP END request. All other flags (`TPNOBLOCK`, `TPNOTIME`, `TPSIGRSTRT`) either are local to the application or affect only client-TEDG interactions in the BEA Tuxedo system. By processing the following flags, the TEDG accomplishes tasks that are normally done in the BEA Tuxedo system.

The following table shows how `tpconnect(3c)` and `tpsend(3c)` flags are mapped.

Table 13-1 BEA Tuxedo Client Flag Mappings

BEA Tuxedo Client Flag	**Action**
`TPNOTRAN`	The TEDG preserves the fact that the request is excluded from the client transaction by the ATMI library.
`TPSENDONLY`	The TEDG returns `TPEV_SVCERR` to the BEA Tuxedo client.
`TPRECVONLY`	This flag must be set to `on`. If it is not, the TEDG returns a `TPEV_SVCERR` event to the BEA Tuxedo client.

A BEA Tuxedo client receives a server response and TEDG errors by calling `tprecv(3c)` in the normal way for a conversation. The TEDG maps the conversation status and error conditions to `tperrno` values or events that are returned with `tprecv()`.

A BEA TOP END server may deliver a response in either a raw buffer or an `FML32` buffer. A raw buffer is normally mapped by the TEDG to a `CARRAY` buffer unless the administrator configures it to map to an `X_OCTET` buffer. Additionally, the administrator may constrain the response buffer to one of three possible types (`CARRAY`, `X_OCTET`, or `FML32`). If the BEA TOP END service returns an incompatible buffer, the TEDG returns a `tperrno` of `TPEOTYPE`.

If `TP_APPL_CONTEXT` is not set on the server response, the TEDG ends the conversation with the BEA Tuxedo client by calling a function that is equivalent to a `tpreturn(3c)` with the `TPSUCCESS` flag set. The client interprets the ending of the conversation as a `TPEV_SVCSUCC` event accompanied by data returned by the server. If `TP_APPL_CONTEXT` is set on the server response, the TEDG maintains the conversation with the BEA Tuxedo client by calling a function that is the equivalent of a `tpsend()` with the `TPRECVONLY` flag set. The client interprets this `tpsend()` call as a `TPEV_SENDONLY` event accompanied by data returned by the server.

When the server returns errors (as a `TP_RESET` status), the TEDG ends the conversation with the BEA Tuxedo client by calling a function that is equivalent to a `tpreturn()` with the `TPFAIL` flag set. The client interprets this call as a `TPEV_SVCFAIL` event. Other errors are returned as a `TPEV_SVCERR` event.

Note: `tpurcode` is not supported by the TEDG.

When a `TPEV_SVCERR` or `TPEV_SVCFAIL` error is returned, the conversation ends. If the server has maintained context, the TEDG disconnects (`TP_DISCONNECT`) the BEA TOP END dialog.

How the TEDG Works with BEA TOP END Servers

Relative to a BEA TOP END server, the TEDG functions as a BEA TOP END client: it receives mapped BEA Tuxedo client conversational requests in the normal manner through `tp_server_receive(3T)`. The buffer received is either a raw buffer or an `FML32` buffer, depending on which buffer types are supported by the BEA Tuxedo client. The BEA TOP END server interprets the message as either a new request or part of a continuing conversation with the BEA Tuxedo client (`TP_APPL_CONTEXT` flag). The BEA TOP END server handles both types of requests according to standard BEA TOP END programming requirements.

For mapped BEA Tuxedo client requests, the `function_qualifier` field cannot be used to indicate one step in a multiple-step conversation. That information must be embedded in the client message.

Client requests may be transactional or non-transactional.

How the TEDG Maps BEA TOP END Server Send Flags

A BEA TOP END server responds to client requests in a normal fashion, using `tp_server_send(3T)`. The response buffer may be either a raw buffer or an `FML32` buffer, depending on which buffer types are supported by BEA TOP END system, the TEDG configuration, and the BEA Tuxedo client. This buffer is mapped to a BEA Tuxedo client buffer. The server ends the conversation by sending a `tp_server_send(3T)` response without the `TP_APPL_CONTEXT` flag.

If a server is to maintain context, the `TP_APPL_CONTEXT` flag in that server is set in the `tp_server_send(3T)` response. This flag instructs the TEDG to continue the conversation and maintain the associated dialog in context mode. The BEA TOP END system routes subsequent messages from the TEDG conversation in the dialog to the same server. The server may indicate an error by resetting the dialog or by responding with an application-defined field value in the response buffer. The BEA Tuxedo client must be programmed to handle that error-reporting interface of the server.

The following table shows how BEA TOP END `tp_server_send(3T)` flags are mapped. The `output_format` and `attach_info` parameters should not be used on responses to the TEDG; they are not supported.

Table 13-2 BEA TOP END Server Send Flag Mapping

BEA TOP END Server Flag	Action
`TP_APPL_CONTEXT` set	Response data is passed to the client with `tpsend(3c)` with the `TPRECVONLY` flag set to switch direction.
`TP_APPL_CONTEXT` not set	Response data is passed to the client and the conversation is ended with `tpreturn(3c)` called with the `TPSUCCESS` flag set.

Table 13-2 BEA TOP END Server Send Flag Mapping (Continued)

BEA TOP END Server Flag	Action
`TP_RESET_DIALOG`	The conversation is ended with `tpreturn(3c)` called with the `TPFAIL` flag set.
`TP_ROLLBACK_ONLY`	The TEDG marks the state of the transaction associated with the dialog as "abort-only."
`TP_DISSOLVE`	Response data is passed to the client and the conversation is ended with `tpreturn(3c)` called with the `TPSUCCESS` flag set.
`TP_FML_BUF`	The TEDG passes the user data as an `FML32` buffer.

When a `TPEV_SVCERR` or `TPEV_SVCFAIL` error is returned, the conversation ends. If the server has maintained context, the TEDG disconnects (`TP_DISCONNECT`) the BEA TOP END dialog.

Error Values

The following error values may be returned to a BEA Tuxedo client because problems exist in the TEDG, the BEA TOP END system, or the BEA TOP END server. Keep in mind that a single error value may be used to report any one of many possible causes of the error being reported.

Because the TEDG does not advertise services based upon the actual availability of BEA TOP END services, a message may be routed to a BEA TOP END node where the services actually are unavailable, resulting in a `TPEV_SVCERR` error, while other routing decisions may result in successful requests. If a service is to be available on multiple nodes, the design of the BEA Tuxedo application, the BEA TOP END application, and the TEDG must take into account the possibility that this type of failure may occur. A well-designed application, that ensures that there are multiple, restartable copies of the servers, reduces the possibility of such errors occurring.

Table 13-3 Error Values Returned to a BEA Tuxedo Client

BEA Tuxedo Error	Cause
TPEV_SVCERR	No match was found on the SERVICE entry.
TPEV_SVCERR	A tpconnect request was sent to a non-conversational service (CONV=N).
TPEV_SVCERR with an error detail of TPED_DOMAINUNREACHABLE	An on-demand connection failed.
TPEV_SVCERR	TP_SERVICE was returned by the BEA TOP END system.
TPEV_SVCERR	The buffer type of the server response does not match the type specified in the OUTBUFTYPE field of the SERVICE entry.
TPEV_SVCERR	The input buffer type does not match the type specified in the INBUFTYPE field of the SERVICE entry or it is not a type supported by the TEDG. (FML32, CARRAY, and X_OCTET are supported.)
TPEV_SVCERR	The size of the input buffer exceeds the maximum message size of 30K bytes (after the FML index is stripped for FML buffers).
TPEV_SVCERR	TPRECVONLY is not specified or TPSENDONLY is set.
TPEV_SVCERR	Error due to public/private key encryption: The client input was rejected because the BEA Tuxedo system is configured to require encryption and the TEDG could not decrypt the message before sending it to the BEA TOP END system.

Table 13-3 Error Values Returned to a BEA Tuxedo Client (Continued)

BEA Tuxedo Error	Cause
`TPEV_SVCERR`	Error due to digital signature: The client input was rejected because the BEA Tuxedo system is configured to require digital signatures and the TEDG could not remove the digital signature from the message before sending the message to the BEA TOP END system.
`TPEV_SVCERR`	All other errors.
`TPEV_SVCFAIL`	`TP_RESET` was returned by the BEA TOP END system.
`TPEV_DISCONIMM`	Link failure, protocol, or state errors occurred.

See Also

- `tpconnect(3c)` in the *BEA Tuxedo ATMI C Function Reference*
- `tpdiscon(3c)` in the *BEA Tuxedo ATMI C Function Reference*
- `tprecv(3c)` in the *BEA Tuxedo ATMI C Function Reference*
- `tpreturn(3c)` in the *BEA Tuxedo ATMI C Function Reference*
- `tpsend(3c)` in the *BEA Tuxedo ATMI C Function Reference*
- `tp_client_send(3T)` in the *BEA TOP END Programmer's Reference Manual*
- `tp_server_receive(3T)` in the *BEA TOP END Programmer's Reference Manual*
- `tp_server_send(3T)` in the *BEA TOP END Programmer's Reference Manual*

How Messages Are Passed from BEA TOP END Clients to BEA Tuxedo Servers

A BEA TOP END client sends requests to BEA Tuxedo servers in the conversational mode in the same way that it sends requests in request/response mode. The TEDG then manages the conversation with the server as shown in the following table.

Table 13-4 Conversational Messaging Process: BEA TOP END Client to BEA Tuxedo Server

If . . .	Then . . .
The `DM_LOCAL_SERVICES` `SERVICE` entry located for this client request is configured as conversational (`CONV=Y`)	The TEDG processes the request as a pseudo-conversation request. For an initial request (before context exists on the client dialog), the TEDG opens a conversation with the server by calling a function equivalent to `tpconnect(3c)` and passes data (if any) and the `TPRECVONLY` flag to the server, thus giving control to the server.
The server responds with a `tpsend(3c)` call with the `TPRECVONLY` flag set	The TEDG responds to the BEA TOP END client with the `TP_APPL_CONTEXT` flag set and keeps the conversation open.
The `TP_DISSOLVE` flag was set by the client	After returning the server's response to the client (`TP_DISSOLVE` and no `TP_APPL_CONTEXT`), the TEDG calls a function equivalent to a `tpdiscon(3c)` to the server.
The `TPRECVONLY` flag is not set by the server, or the flags are set to `TPSENDONLY` **Note:** The mapping of BEA TOP END pseudo-conversations to BEA Tuxedo conversations requires that the BEA Tuxedo server relinquish control of the conversation with each response.	The TEDG logs an error and ends the conversation using `tpdiscon(3c)`. The BEA TOP END dialog is reset (`TP_RESET`).

Table 13-4 Conversational Messaging Process: BEA TOP END Client to BEA Tuxedo Server (Continued)

If . . .	Then . . .
The server returns by ending the conversation (`tpreturn(3c)`) with `TPSUCCESS`	The TEDG responds to the BEA TOP END client without setting the `TP_APPL_CONTEXT` flag.
The server returns by ending the conversation (`tpreturn(3c)`) with `TPFAIL` or `TPEXIT`	The TEDG responds by resetting the dialog that returns `TP_RESET` with an extended status of `TP_EXT_SERVER_APPL` to the BEA TOP END client.
The client continues context by specifying a blank product and blank function parameter	A client request that continues the BEA TOP END application context on the dialog is sent to the same conversation with a function equivalent to `tpsend(3c)` with the `TPRECVONLY` flag set. **Note:** With TEDG-mapped conversations, the BEA TOP END client cannot use the `function_qualifier` to specify a step in a multiple-step interaction. This information must be passed in the client message.
The client breaks the context by performing a function switch (specifying a product and function)	The TEDG receives a `TP_DISCONNECT` and terminates the conversation using a function equivalent to `tpdiscon(3c)` and acknowledges the disconnected status to the BEA TOP END system.

How the TEDG Works with BEA TOP END Clients

Client operations are programmed with the same functions used for any BEA TOP END client.

Use This API . . .	For . . .
`tp_client_send`	Making asynchronous requests

Use This API . . .	For . . .
`tp_client_signon`	Making asynchronous requests
`tp_client_receive`	Receiving responses

These functions are used in the normal manner when making a service request to a BEA Tuxedo server through the TEDG.

As a BEA TOP END client programmer, you need to know the following information:

- The product and function name assigned by the administrator, in the `DMCONFIG` file, to the BEA Tuxedo service.
- If you are trying to match existing BEA TOP END programming (unlikely in the case of conversational messaging), the function qualifier value, optionally assigned by the administrator.
- The MSR target name, which may have been assigned by the administrator in the `DMCONFIG` file. (Assigning an MSR target is optional.) This MSR target name may be used:
  - To match a configured BEA TOP END MSR routing strategy, or
  - Directly by the BEA TOP END client application.
- The buffer type required by the BEA Tuxedo service. A raw message must be sent if the BEA Tuxedo server requires either `CARRAY` or `X_OCTET`. By default a raw message is mapped to a `CARRAY` buffer. Buffers of these types have no data marshalling support when transmitted to the TEDG by a BEA TOP END node of a different type.

  If a BEA Tuxedo service supports `FML32` input, the BEA TOP END client must use the `FML32` message type. `FML32` is beneficial because the TEDG and the BEA TOP END system support data marshalling of these buffers when the buffers are transmitted between the TEDG and a BEA TOP END node of a different type.

  A BEA Tuxedo service may support one or more of these buffer types.
- Whether the administrator has required, in the `DMCONFIG` file, the exclusive use of one buffer type. If a buffer of a type that is inappropriate (due to the fact that it does not match either the configured type or any type supported by the TEDG) is sent to the TEDG, a `tp_client_receive` call returns a `TP_RESET` error.

How the TEDG Maps Client Requests

A client request may be transactional or non-transactional; it must require a response. The following table shows how BEA TOP END client flags are mapped. By mapping these flags, the TEDG accomplishes a task that is normally done in the BEA TOP END system. Do not use the `input_format` and `attach_info` parameters on requests to the TEDG; they are not supported.

Table 13-5 BEA TOP END Client Flag Mapping

BEA TOP END Client Flag	Action
`TP_DISSOLVE`	The TEDG dissolves the dialog and returns the server response. It then ends the conversation by issuing a `tpdiscon(3c)` call.
`TP_NON_TRANSACT`	The TEDG preserves the fact that the request is excluded from the client transaction by the CSI library.
`TP_NO_RESPONSE`	This flag is not supported for pseudo-conversations. The TEDG resets the dialog and returns `TP_RESET` with an extended status of `TP_EXT_SERVER_APPL`.
`TP_FML_BUF`	The TEDG passes the user data as an `FML32` buffer.

The status and extended status values returned to the BEA TOP END client are standard values. For additional information about the mapping of error values, see “Error Values” on page 13-15.

The TEDG maps the response from the BEA Tuxedo system or server to a response that the BEA TOP END client accesses through the `tp_client_receive(3T)` call. BEA Tuxedo servers should send responses in one of the following types of buffers: `CARRAY`, `X_OCTET`, or `FML32`. A `CARRAY` or `X_OCTET` buffer is mapped by the TEDG to a raw message; an `FML32` buffer, to a BEA TOP END `FML32` message. The administrator may constrain the response buffer to a specific type (`CARRAY`, `X_OCTET`, or `FML32`). If the BEA Tuxedo service returns an incompatible buffer, then the TEDG returns a `TP_RESET` status.

How the TEDG Works with BEA Tuxedo Servers

Relative to a BEA Tuxedo server, the TEDG functions as a conversational BEA Tuxedo client: it receives mapped BEA TOP END client requests in the normal manner. The TEDG always relinquishes control after one message is sent. As a result:

- The server is notified of the `TPEV_SENDONLY` event (`tprecv`) or
- `TPSVCINFO`->flags have `TPSENDONLY` on the initial message and invocation of the server.

The buffer received is a `CARRAY`, `X_OCTET`, or `FML32` buffer, depending on the message sent by the client. The BEA Tuxedo server processes the request in the normal manner. Client requests may be transactional or non-transactional.

A BEA Tuxedo server responds to client requests in the standard manner. If the conversation is being continued, it calls `tpsend(3c)` with the `TPRECVONLY` flag and data. In this case the server must then call `tprecv(3c)` to receive the next client message, an error indication, or an indication that the conversation was terminated (`TPEV_DISCONIMM`). To send the last message of a conversation, the server calls `tpreturn(3c)` with the `TPSUCCESS` flag. To terminate a conversation and indicate an error, a server calls `tpreturn()` with the `TPFAIL` flag. Reply messages are supported only with `tpsend` or `TPSUCCESS`. The application-defined return code, *`rcode`*, is not supported by the TEDG.

The BEA Tuxedo server buffer may be a `CARRAY`, `X_OCTET,` or `FML32` buffer, depending on which types are supported by the BEA TOP END system, the TEDG configuration, and the BEA TOP END client. This buffer is mapped to a BEA TOP END message as described in "How the TEDG Maps Client Requests" on page 13-13. The server may indicate an error by calling `TPFAIL` or by responding with an application-defined field value in the reply buffer. The BEA TOP END client must be programmed accordingly. To terminate a conversation, the client ends the dialog by calling `tp_client_signoff(3T)` or invoking a function switch that calls a different service.

Because BEA TOP END messages are limited to 30K bytes, the BEA Tuxedo server reply may not exceed that limit. For an `FML32` message, the limit applies after the FML index is stripped from the message.

How the TEDG Maps BEA Tuxedo Server Flags

The following table shows how the TEDG maps flags to `tpsend(3c)`, the function that transmits BEA Tuxedo server responses. All other flags (`TPNOBLOCK`, `TPNOTIME`, `TPSIGRSTRT`) either are local to the application or affect only server-TEDG interactions in the BEA Tuxedo system.

Table 13-6 BEA Tuxedo Server Flag Mapping

BEA Tuxedo Server Flag	Action
`TPRECVONLY`	This flag must be set `on`. If it is not, the TEDG resets the dialog and returns `TP_RESET` with an extended status of `TP_EXT_SERVER_APPL`.
`TPSENDONLY`	The TEDG resets the dialog and returns `TP_RESET` with an extended status of `TP_EXT_SERVER_APPL`.

Error Values

In addition to regular BEA TOP END error status messages, a number of other status messages may be returned to a BEA TOP END client as a result of problems in the TEDG, the BEA Tuxedo system, or the BEA Tuxedo server. Keep in mind that a single error status value may be used to report any one of many possible causes of the error being reported.

Because the TEDG does not advertise services based upon the actual availability of BEA Tuxedo services, a message may be routed to a BEA Tuxedo node where the services actually are unavailable, resulting in a `TP_SERVICE` error, while other routing decisions may result in successful requests. If a service is to be available on multiple nodes, the design of the BEA Tuxedo application, the BEA TOP END application, and the TEDG must take into account the possibility that this type of failure may occur. A well-designed application, that ensures that there are multiple, restartable copies of the servers, reduces the possibility of such errors occurring.

Table 13-7 Error Values Returned to a BEA TOP END Client

BEA TOP END Error Status	Cause
TP_SERVICE, TP_EXT_MSR_FAILURE	The TEDG SERVICE entry lookup failed: the target was not found.
TP_SERVICE, TP_EXT_NO_SUCH_SERV	The TEDG SERVICE entry lookup failed: the product or function was not found (ignoring entries for which RDOM does not have access).
TP_SERVICE, TP_EXT_NO_SUCH_SERV	TPENOENT was returned by the BEA Tuxedo system.
TP_RESET, TP_EXT_SERVER_APPL	The input buffer type does not match the type specified in the INBUFTYPE field of the SERVICE entry.
TP_RESET, TP_EXT_SERVER_APPL	The server response buffer type does not match the type specified in the OUTBUFTYPE field of the SERVICE entry or is not a type supported by the TEDG. (FML32, CARRAY, and X_OCTET are supported.)
TP_RESET, TP_EXT_SERVER_APPL	The size of the server response buffer exceeds the maximum message size of 30K bytes (after the FML index is stripped for FML buffers).
TP_RESET, TP_EXT_SERVER_APPL	Server failed and returned TPFAIL.
TP_RESET, TP_EXT_SERVER_APPL	TPRECVONLY is not specified or TPSENDONLY is set on server response.
TP_RESET, TP_EXT_SERVER_APPL	All other tperrno values.
TP_RESET, TP_EXT_SERVER_APPL	Error due to public/private key encryption: The client input was rejected by the BEA Tuxedo server because the BEA Tuxedo system is configured to require encryption and the TEDG could not decrypt the message before sending it to the BEA TOP END system.

Table 13-7 Error Values Returned to a BEA TOP END Client (Continued)

BEA TOP END Error Status	Cause
`TP_RESET`, `TP_EXT_SERVER_APPL`	Error due to digital signature: The client input was rejected by the BEA Tuxedo server because the BEA Tuxedo system is configured to require digital signatures and the TEDG could not remove the digital signature from the message before sending the message to the BEA TOP END system.
`TP_RESET`, `TP_EXT_SERVER_FAIL`	Internal error occurred within the TEDG.

See Also

- `tpconnect(3c)` in the *BEA Tuxedo ATMI C Function Reference*
- `tpdiscon(3c)` in the *BEA Tuxedo ATMI C Function Reference*
- `tprecv(3c)` in the *BEA Tuxedo ATMI C Function Reference*
- `tpreturn(3c)` in the *BEA Tuxedo ATMI C Function Reference*
- `tpsend(3c)` in the *BEA Tuxedo ATMI C Function Reference*
- `tp_client_receive(3T)` in the *BEA TOP END Programmer's Reference Manual*
- `tp_client_send(3T)` in the *BEA TOP END Programmer's Reference Manual*
- `tp_client_signon(3T)` in the *BEA TOP END Programmer's Reference Manual*

CHAPTER

14 Reliable Queuing Programming

This topic includes the following sections:

- Using Reliable Queuing with the TEDG
- How BEA Tuxedo Clients Enqueue Messages to RTQ
- How BEA TOP END Clients Enqueue Messages to /Q

Using Reliable Queuing with the TEDG

The BEA Tuxedo and BEA TOP END systems support reliable queues that can pass messages between components. The BEA Tuxedo /Q facility and the BEA TOP END Recoverable Transaction Queuing (RTQ) facility allow messages to be passed using queues to store messages before processing. Both /Q and RTQ guarantee that once a message is successfully placed on a queue, it will be delivered to the server. Full transaction semantics are supported for both queuing and processing the message.

The TOP END Domain Gateway (TEDG) supports the enqueuing of messages between BEA Tuxedo and BEA TOP END systems. Transactional enqueuing of messages between systems is also supported. The queue itself is part of the native system on which it was created. All the administrative aspects of managing the queue and dequeuing messages are also part of the native system.

The TEDG does not support dequeuing from a queue in a remote system because the /Q and RTQ capabilities and interfaces on the dequeuing side are quite different. BEA Tuxedo /Q provides the `tpdequeue(3c)` function for explicitly dequeuing a message; there is no equivalent within RTQ. The RTQ facility dequeues messages automatically and delivers them to the intended BEA TOP END service. The destination service address is supplied as part of enqueuing the message (see `tp_rtq_put(3T)` in the *BEA TOP END Programmer's Reference Manual*).

The relationship between the queues and the services is arbitrary. /Q provides a system-supplied service, `TMQFORWARD(5)`, that can be configured to dequeue messages automatically and forward them to standard BEA Tuxedo servers through the `tpcall(3c)` function. The destination service name must match the queue name. In a TEDG environment, the server to which these messages are sent after automatic dequeuing may be in the same system or a remote system, depending on the TEDG configuration and appropriate server programming.

We assume that you already have a BEA Tuxedo /Q queue or a BEA TOP END RTQ queue and the administrative tools and servers associated with the queue. The *Using the ATMI /Q Component* and the *BEA TOP END Recoverable Transaction Queuing Guide* describe the value of queuing, discuss the task of designing a system in which queuing is used, and explain how to set up and manage queues.

Common Queuing Capabilities Supported by the TEDG

While /Q and RTQ offer similar basic capabilities, each also offers several unique features. The capabilities offered by the TEDG represent the common subset of the features of the two systems:

- Non-transactional intersystem queuing
- Transactional intersystem queuing
- Scheduling time options

Unsupported BEA Tuxedo /Q Capabilities

The TEDG does not support the following BEA Tuxedo /Q features:

- Priorities

- Reply queue and failure queue
- Expiration time
- Quality of service
- Explicit dequeue (from an RTQ queue)

Unsupported BEA TOP END RTQ Capabilities

The TEDG does not support the following BEA TOP END RTQ features:

- Non-transactional scheduling
- Transaction key scheduling
- Initial held message status
- Tag text
- Large Message Architecture (LMA)

See Also

- “BEA Tuxedo /Q C Language Programming” on page 3-1 in *Using the ATMI /Q Component*
- *BEA TOP END Recoverable Transaction Queuing Guide*

How BEA Tuxedo Clients Enqueue Messages to RTQ

BEA Tuxedo clients may enqueue messages to BEA TOP END RTQ queues by calling `tpenqueue(3c)`.The BEA Tuxedo application routes the request to the TEDG based on the queue space (`QSPACE`) parameter specified in the call. The TEDG appears to the BEA Tuxedo client as if it is a `TMQUEUE(5)` server.

The BEA TOP END administrator must create an RTQ queue in the BEA TOP END system. (For details on creating an RTQ queue, see the *BEA TOP END Recoverable Transaction Queuing Guide*.) To advertise an RTQ queue as a BEA Tuxedo queue space, the TEDG must have a `QSPACE` entry for it in the `DM_REMOTE_SERVICES` section of the `DMCONFIG` file. The status of the RTQ queue in the BEA TOP END system (that is, whether the RTQ server for the queue is available) is not tracked while the connection is active. These TEDG-supported queue spaces are not defined using `qmadmin`, as queue spaces are defined for BEA Tuxedo /Q queues.

The TEDG uses the `TE_RTQGROUP`, `TE_RTQNAME`, and optional `TE_TARGET` parameters in the queue space entry to determine the corresponding BEA TOP END queue. The TEDG uses the queue name (`qname`) parameter from the `tpenqueue()` function to determine the name of the service for the RTQ request. The `DM_REMOTE_SERVICES` section is searched for a `QNAME` entry that matches the queue name. The values of four associated parameters—`TE_PRODUCT`, `TE_FUNCTION`, `TE_TARGET,` and `TE_QUALIFIER`—are retrieved and included in the message sent to the BEA TOP END system. To the BEA TOP END system, the TEDG appears to be making a `tp_rtq_put` request.

The TEDG returns `TPENOENT` if the queue space cannot be mapped successfully. A `tpenqueue()` return code of `TPEDIAGNOSTIC` and a diagnostic value of `QMEBADQUEUE` are returned if `qname` cannot be mapped. The status returned by the BEA TOP END system is mapped to a BEA Tuxedo return value and sent to the BEA Tuxedo client.

The message enqueued to the RTQ queue is scheduled by RTQ and the recipient server accesses the message in the standard way. The client identifier associated with the request is the TEDG local domain ID. As with all RTQ messages, the server responds to RTQ upon completion but it cannot return data. If the server needs to reply, the client and server must pass reply queue information within the actual client message to emulate the way that replies are handled by the /Q feature of the BEA Tuxedo system.

How the TEDG Works with BEA Tuxedo Clients

A BEA Tuxedo client programmer uses the `tpenqueue()` function to enqueue a message to a BEA TOP END RTQ queue using the TEDG.

As a BEA Tuxedo client programmer, you need to know the following information:

- The queue space name assigned by the administrator, in the `DMCONFIG` file, to the BEA TOP END RTQ queue.
- The queue name (`qname`) assigned to the BEA TOP END service to which the message will be sent at the time scheduled by RTQ.
- The buffer type required by the BEA TOP END service. A BEA Tuxedo buffer type of `CARRAY` or `X_OCTET` is used to prepare the raw buffer required by the BEA TOP END service. Buffers of these types have no data marshalling support when transmitted by the TEDG between BEA TOP END nodes of different types.

  If a BEA TOP END service supports `FML32` input, the BEA Tuxedo client uses the BEA Tuxedo `FML32` buffer type. `FML32` is beneficial because the TEDG and the BEA TOP END system support data marshalling of these buffers when the buffers are transmitted between the TEDG and a BEA TOP END node of a different type.

  A BEA TOP END service may support one or more of these buffer types.
- Whether the administrator has required, in the `DMCONFIG` file, the exclusive use of one buffer type. If a buffer of a type that is inappropriate (due to the fact that it does not match either the configured type or any type supported by the TEDG) is sent to the TEDG, a call to `tpenqueue()` returns `TPEDIAGNOSTIC` and `QMEINVAL` errors.

Because BEA TOP END RTQ messages are limited to 30,000 bytes, the client request may not exceed that limit. For `FML32` messages, the limit applies after the FML index is stripped from the message.

How the TEDG Maps Client Requests

A client request may be transactional or non-transactional. The following table shows how the BEA Tuxedo client `tpenqueue` flags and associated parameters are mapped to a BEA TOP END RTQ request. By mapping the following flags and parameters, the TEDG performs a task that is normally done in the BEA Tuxedo system.

Table 14-1 BEA Tuxedo Client Flag Mapping

BEA Tuxedo Client Flag	Action
`TPQTIME_ABS`	If set, the value in `TPQCTL deq_time` is the actual time the client wishes to have the message dequeued. The value is stored as a UNIX time type. The TEDG passes this time value to the RTQ server to be carried out as a client-relative, absolute-time value.
`TPQTIME_REL`	If set, the value in `TPQCTL` deq_time is the amount of time, in seconds, the client wishes to wait before the message can be dequeued. The TEDG maps this value to the RTQ `schedule_time` parameter (in `R:HH:MM` format).
`TPNOTRAN`	The TEDG preserves the fact that the `tpenqueue` request is excluded from the client transaction by the ATMI library.
`TPQMSGID`	If set, the RTQ `request_id` returned by RTQ is stored in the `TPQCTL msgid` field on a successful request.

All other `tpenqueue()` option flags, including those in the following list, are not supported by the TEDG. The `urcode` field in `TPQCTL` is also not supported.

- `TPQTOP`
- `TPQBEFOREMSGID`
- `TPQPRIORITY`
- `TPQREPLYQ`
- `TPQFAILUREQ`

- TPQCORRID
- TPQDELIVERYQOS
- TPQREPLYQOS
- TPQEXPTIME_ABS
- TPQEXPTIME_REL
- TPQEXPTIME_NEVER
- TPQOSDEFAULTPERSIST
- TPQOSPERSISTENT
- TPQOSNONPERSISTENT

The `tperrno` values returned to the BEA Tuxedo client on the `tpenqueue()` call are standard values. Because the TEDG acts as a `TMQUEUE` server, it maps many TEDG and RTQ related errors to both the `TPEDIAGNOSTIC` `tperrno` and a corresponding value for the `TPQCTL` diagnostic field.

Error Values

The following error values may be returned to a BEA Tuxedo client because problems exist in the TEDG, the BEA TOP END system, or the BEA TOP END server. Keep in mind that a single error value may be used to report any one of many possible causes of the error being reported.

Because the TEDG does not advertise the `QSPACE` based upon the actual availability of the BEA TOP END RTQ server that handles the queue, a message may be routed to a BEA TOP END node where the queue space actually is unavailable, resulting in a `tperrno` of `TPENOENT`, while other routing decisions may result in successful requests. If a queue space is to be available on multiple nodes, the design of the BEA Tuxedo application, the BEA TOP END application, and the TEDG must take into account the possibility that this type of failure may occur. A well-designed application, that ensures that there are multiple, restartable copies of the servers, reduces the possibility of such errors occurring.

Table 14-2 Error Values Returned to a BEA Tuxedo Client

BEA Tuxedo Error	Cause
TPENOENT	No match was found on a QSPACE entry for the QSPACE parameter.
TPENOENT	The BEA TOP END system returned TP_SERVICE; the RTQ server for the queue is unavailable.
TPENOENT	An on-demand connection failed.
TPESYSTEM	Error due to public/private key encryption: The client input was rejected because the BEA Tuxedo system is configured to require encryption and the TEDG could not decrypt the message before sending it to the BEA TOP END system.
TPESYSTEM	Error due to digital signature: The client input was rejected because the BEA Tuxedo system is configured to require digital signatures and the TEDG could not remove the digital signature from the message before sending the message to the BEA TOP END system.
TPEDIAGNOSTIC, QMEBADQUEUE	No match was found on a QNAME entry for qname parameter.
TPEDIAGNOSTIC, QMEINVAL	One of these unsupported flags was set: TPQREPLYQ, TPQFAILUREQ, TPQCORRID, TPQBEFOREMSGID, TPQTOP, TPQPRIORITY, TPQDELIVERYQOS, TPQREPLYQOS, TPQEXPTIME_ABS, TPQEXPTIME_REL, TPQEXPTIME_NEVER, TPQOSDEFAULTPERSIST, TPQOSPERSISTENT, TPQOSNONPERSISTENT

Table 14-2 Error Values Returned to a BEA Tuxedo Client (Continued)

BEA Tuxedo Error	Cause
`TPEDIAGNOSTIC`, `QMEINVAL`	The user buffer type does not match the type specified in the `INBUFTYPE` field of the `QNAME` entry or it is not a type supported by the TEDG. (`FML32`, `CARRAY`, and `X_OCTET` buffer types are supported.)
`TPEDIAGNOSTIC`, `QMEINVAL`	The user buffer exceeds the maximum RTQ message size of 30,000 bytes (after the FML index is stripped for FML buffers).
`TPEDIAGNOSTIC`, `QMEINVAL`	The `TPQTIME_REL` flag is set and the value in `TPQCTL deq_time` is set to be greater than 86400 (that is, 24 hours).
`TPEDIAGNOSTIC`, `QMENOSPACE`	The BEA TOP END system returned `TP_RTQ_EOF` and `TP_RTQ_NO_SPACE`.
`TPEDIAGNOSTIC`, `QMESYSTEM`	The BEA TOP END system returned `TP_RTQ_ERROR`, `TP_RTQ_MEMERR`, and `TP_RTQ_QDISABLED`.
`TPEDIAGNOSTIC`, `QMESYSTEM`	The BEA TOP END system returned `TP_RESET`.

See Also

- `tpenqueue(3c)` in the *BEA Tuxedo ATMI C Function Reference*

How BEA TOP END Clients Enqueue Messages to /Q

BEA TOP END clients enqueue messages to BEA Tuxedo /Q queues by calling `tp_rtq_put(3T)`. The `queue_info` parameter is used by the BEA TOP END system to route the request to the TEDG or another RTQ server. The `queue_info` parameter specifies the RTQ group, RTQ queue name, and RTQ target.

The BEA Tuxedo administrator must create the /Q queue space and create the queue names available within that `QSPACE` using `qmadmin(1)`. To make the BEA Tuxedo /Q queue space available to the BEA TOP END system, a `QSPACE` entry for the queue space must be configured in the `DM_LOCAL_SERVICES` section of the `DMCONFIG` file and the `TE_RTQGROUP`, `TE_RTQNAME`, and `TE_TARGET` parameters must be specified. The TEDG advertises these parameters when a connection to the BEA TOP END system is established. The status of individual queue space availability in the BEA Tuxedo domain is not tracked while the connection is active.

When a request is received by the TEDG, the `queue_info` parameter (which defines the RTQ group, queue name, and target) is used to determine the proper queue space. If a target is not specified, none is included in the lookup. The `tp_rtq_put` service parameter is used to determine the /Q queue name. The TEDG searches the `DM_LOCAL_SERVICES` section looking for a `QNAME` entry that matches the product, function, target, and function qualifier from the message. The TEDG enqueues the message using the derived queue space and queue name parameters by calling a function equivalent to `tpenqueue(3c)`. Default values are used as parameters to the `tpenqueue` options. For example, priority cannot be mapped from an RTQ request, so the default is used. After the message is enqueued, the BEA Tuxedo administrator may use `qmadmin` to modify attributes of the /Q queue and the messages in it.

The TEDG maps the status returned by the `TMQUEUE(5)` server and returns it to the BEA TOP END client. On successful enqueue requests, the TEDG assigns a unique RTQ `request_id` that is returned to the client. The `request_id` is provided only for tracking the status of requests; it cannot be used in any further administration of the request.

The recipient of the message enqueued to the /Q queue accesses the message in the standard way, depending on whether `tpdequeue(3c)` is called or `TMQFORWARD(5)` is used to turn the message into a service request. The buffer type received is determined by the message type sent and the `DMCONFIG` parameters. Fields on the `TPQCTL` structure are set to appropriate values, but features such as priority, correlation ID, reply queue, failure queue, and user-return code are not supported by the TEDG; they are either not set or they are set to default values. The `appkey` and `cltid` fields are set with the BEA Tuxedo user ID (the `DOMAINID` of the remote domain) of the client that sent the message.

How the TEDG Works with BEA TOP END Clients

The BEA TOP END client programmer uses the `tp_rtq_put(3T)` call to enqueue a message to a BEA Tuxedo /Q queue using the TEDG.

As a BEA TOP END client programmer, you need to know the following information:

- The RTQ group, RTQ queue name, and RTQ target assigned to the BEA Tuxedo /Q queue. These parameters are defined by the administrator in the `QSPACE` entry in the `DM_LOCAL_SERVICES` section of the `DMCONFIG` file. These values are then specified in the `queue_info` parameter of the `tp_rtq_put(3T)` function.
- The product and function name for the BEA Tuxedo queue name for which the message is destined. These are assigned by the administrator in a `QNAME` entry in the `DM_LOCAL_SERVICES` section of the `DMCONFIG` file.
- If you are trying to match existing BEA TOP END programming, the function qualifier value, optionally assigned by the administrator.
- The MSR target name which may have been assigned by the administrator in the `DMCONFIG` file. (Assigning a name to the MSR target is optional). This name may be used:
  - To match a configured BEA TOP END MSR routing strategy, or
  - Directly by the BEA TOP END client application.
- The buffer type required by the recipient BEA Tuxedo service. If the BEA Tuxedo server requires either `CARRAY` or `X_OCTET`, a raw message must be sent. By default a raw message is mapped to a `CARRAY` buffer.

  `CARRAY` and `X_OCTET` buffer types have no data marshalling support when transmitted to the TEDG by a BEA TOP END node of a different type. If a BEA Tuxedo service supports `FML32` input, the BEA TOP END client uses the BEA Tuxedo `FML32` buffer type. `FML32` is beneficial because the TEDG and the BEA

TOP END system support data marshalling of these buffers when the buffers are transmitted between the TEDG and a BEA TOP END node of a different type.

A BEA TOP END service may support one or more of these buffer types.

- Whether the administrator has required, in the `DMCONFIG` file, the exclusive use of one buffer type. If a buffer of a type that is inappropriate (due to the fact that it does not match either the configured type or any type supported by the TEDG) is sent to the TEDG, a call to `tp_rtq_put()` returns a `TP_RTQ_PARAMERR` error.

To make a `tp_rtq_put` request, you must specify the product, function, MSR target (optional), and function qualifier (optional) associated with the desired BEA Tuxedo queue name in the *`service`* parameter.

How the TEDG Maps Client RTQ Requests

A client request may be transactional or non-transactional. The following table shows how BEA TOP END client RTQ flags and parameters are mapped. By mapping these flags and parameters, the TEDG performs a task that is normally done in the BEA TOP END system. The `TP_RTQ_HELD` and `TP_RTQ_NON_TRANSACT_SCHED` flags, the transaction key feature, and the `tag_length`, `tag_text`, `input_format`, and `attach_info` parameters should not be used on requests to the TEDG; they are not supported.

Table 14-3 BEA TOP END Client RTQ Flag and Parameter Mapping

RTQ Flag or Parameter	Action
`TP_RTQ_NON_TRANSACT_QUEUE`	The TEDG preserves the fact that the RTQ request is excluded from the client transaction by the CSI library.
`schedule_time` (absolute, with respect to the server time zone in *`HH:MM`* format)	The value of the `schedule_time` parameter forms the hour and minute portion of the `TPQCTL` `deq_time` variable. If `schedule_time` has passed, a request is scheduled at `schedule_time` plus 1 day. The `TPQTIME_ABS` flag is set on. **Note:** The absolute time is mapped with respect to the TEDG time zone rather than the time on the /Q `TMQUEUE(5)` server. The TEDG time zone is used because `tpenqueue(3c)` does not offer an equivalent option for specifying time with respect to the time zone of the server.

Table 14-3 BEA TOP END Client RTQ Flag and Parameter Mapping

RTQ Flag or Parameter	Action
`schedule_time` (relative in *`R:HH:MM`* format)	Mapped to the `TPQCTL` `deq_time`, this parameter converts the `schedule_time` from *`R:HH:MM`* format to the number of seconds to delay dequeuing. The `TPQTIME_REL` flag is set `on`.
`schedule_time` (with respect to the client time zone in *`L:HH:MM`* format)	Mapped to the `TPQCTL` `deq_time`. The `TPQTIME_ABS` flag is set `on`.
`TP_RTQ_FML_BUF`	The TEDG passes user data, as an FML buffer, in a `tpenqueue(3c)` call.

Error Values

In addition to regular BEA TOP END error status messages, a number of other status messages may be returned to a BEA TOP END client as a result of problems in the TEDG, the BEA Tuxedo system, or the BEA Tuxedo `TMQUEUE` server. Keep in mind that a single error status message may be used to report any one of many possible causes of the error being reported.

Because the TEDG does not advertise the RTQ queue based upon the actual availability of BEA Tuxedo queue space, a message may be routed to a BEA Tuxedo node where queue space actually is unavailable, resulting in an error status of `TP_RTQ_UNAVAIL`, while other routing decisions may result in successful requests. If an RTQ queue is to be available on multiple nodes, the design of the BEA Tuxedo application, the BEA TOP END application, and the TEDG must take into account the possibility that this type of failure may occur. A well-designed application, that ensures that there are multiple, restartable copies of the servers, reduces the possibility of such errors occurring.

Table 14-4 Error Values Returned to a BEA TOP END RTQ Client

BEA TOP END Error Status	Cause
`TP_RTQ_UNAVAIL`, `TP_RTQ_EXT_MSR_FAILURE`	The TEDG `QSPACE` entry lookup failed; the target was not found.

Table 14-4 Error Values Returned to a BEA TOP END RTQ Client (Continued)

BEA TOP END Error Status	Cause
`TP_RTQ_UNAVAIL`, `TP_RTQ_EXT_NOT_AVALIABLE`	The TEDG `QSPACE` entry lookup failed; the RTQ group and queue names were not found. (Entries for services to which `RDOM` clients do not have access were ignored.)
`TP_RTQ_UNAVAIL`, `TP_RTQ_EXT_NOT_AVAILABLE`	`TPENOENT` was returned by the BEA Tuxedo system.
`TP_RTQ_PARAMERR`	The TEDG `QNAME` entry lookup failed.
`TP_RTQ_PARAMERR`	The input buffer type does not match the type specified in the `INBUFTYPE` parameter for the `SERVICE` entry.
`TP_RTQ_PARAMERR`	Unsupported RTQ flags `TP_RTQ_HELD` and `TP_RTQ_NON_TRANSACT_SCHED` were used.
`TP_RTQ_PARAMERR`	The BEA TOP END RTQ client specified a transaction key greater than zero; this is unsupported.
`TP_RTQ_PARAMERR`	The BEA Tuxedo `TMQUEUE` server returned `TPEDIAGNOSTIC`, `QMEBADQUEUE` because the queue name was not defined for the queue space. The queue name and queue space were determined by mapping the RTQ client request.
`TP_RTQ_EOF`	The BEA Tuxedo `TMQUEUE` server returned `TPEDIAGNOSTIC`, `QMENOSPACE` because there was no space on the queue for the message.
`TP_RTQ_ERROR`	All other errors.

See Also

- `qmadmin(1)` in the *BEA Tuxedo Command Reference*
- `tpdequeue(3c)` in the *BEA Tuxedo ATMI C Function Reference*
- `tpenqueue(3c)` in the *BEA Tuxedo ATMI C Function Reference*

- `tp_rtq_put(3T)` in the *BEA TOP END Programmer's Reference Manual*

CHAPTER

15 Transactional Support Programming

This topic includes the following sections:

- Using Transactions with the TEDG
- Transaction Capabilities Supported by the TEDG

Using Transactions with the TEDG

The TOP END Domain Gateway (TEDG) provides support for fully transactional message passing and queuing between BEA TOP END and BEA Tuxedo systems. Transactions may be initiated by either system.

The TEDG does the following:

- Serves as the transaction coordinator between the systems and adjusts to the requirements of each system.
- Maps transaction IDs and logs the state of transactions as required for recovery.
- On recovery after a failure in a TEDG node or TEDG process, the TEDG examines the transaction log and takes action to complete transactions that have reached the “prepared” state. Any transactions that are still in progress are aborted or rolled back.

Transaction Capabilities Supported by the TEDG

The following transaction capabilities are supported:

- Transactions may span both BEA TOP END and BEA Tuxedo Application-to-Transaction Monitor Interface (ATMI) applications.
- Transactions may be started by either the BEA TOP END or BEA Tuxedo application.
- Transactions are supported for request/response and conversational message passing.
- Transactions are supported for enqueuing messages to queues. Because dequeuing does not involve the TEDG, the transactional characteristics for dequeuing messages are unchanged; they are specific to each system.
- Transactions may involve one or more local components and have one or more transaction branches into the remote system through the TEDG.
- Transactions abide by local system conventions with respect to abort/rollback due to message passing or queuing errors before the application issues a commit or abort/rollback.
- Transactions may infect a remote system and then reinfect the original system. For example, a BEA Tuxedo client may send a transactional request to a BEA TOP END server, which then sends a transactional request to a BEA Tuxedo server.

Transaction Design Consideration

From the programmer’s perspective, transactions involving the TEDG and a remote system function are handled the same way any transactions are handled in the local system. The only consideration to keep in mind when designing transactions for a configuration that includes BEA Tuxedo and BEA TOP END systems is that two-phase commit processing cannot be optimized as much as it can be optimized for transactions that involve only one system. Therefore, transactional requests involving a remote system have a slightly higher cost to commit. When designing your application, you should consider the number of intersystem requests likely to be made and the potential effect of these requests on performance.

See Also

- "Writing Global Transactions" on page 9-1 in *Programming BEA Tuxedo ATMI Applications Using C*
- *BEA TOP END Application Programmer's Guide*

CHAPTER

16 Security Programming

This topic includes the following sections:

- How the TEDG Supports Security
- How BEA Tuxedo Client Requests Are Authorized
- How to Establish Security for BEA TOP END Services/RTQ Queues
- How BEA TOP END Client Requests Are Authorized
- How to Establish Security for BEA Tuxedo Services/Queue Spaces
- How Security Is Provided for the TEDG Network Connection

How the TEDG Supports Security

Both the BEA TOP END and BEA Tuxedo systems offer integrated security services. The TEDG works with both sets of services. Authentication of client applications is the responsibility of the local system and is performed in the same way for all configurations, regardless of whether the TEDG is included.

The only consideration to keep in mind when programming security for a configuration that includes a TEDG is that different functions are used in the BEA Tuxedo and BEA TOP END systems to specify client identifiers and passwords to the local programming interfaces. In BEA Tuxedo code, this task is done through the `tpinit(3c)` function; in BEA TOP END code, through the `tp_client_signon(3T)`, `tp_rtq_signon(3T)`, and `tp_rtq_put(3T)` calls. In a BEA Tuxedo application, one ID is used for a client program; in a BEA TOP END application, a separate client ID is associated with each dialog. For an administrator's view of security administration on TEDG-based configurations, refer to "Configuring Security Between BEA TOP END and BEA Tuxedo Systems" on page 7-1.

See Also

- "Understanding Security Between the BEA TOP END and BEA Tuxedo Systems" on page 3-1
- `tpinit(3c)` in the *BEA Tuxedo ATMI C Function Reference*
- `tp_client_signon(3T)` in the *BEA TOP END Programmer's Reference Manual*
- `tp_rtq_put(3T)` in the *BEA TOP END Programmer's Reference Manual*
- `tp_rtq_signon(3T)` in the *BEA TOP END Programmer's Reference Manual*

How BEA Tuxedo Client Requests Are Authorized

Authorization of client requests is based on each system's local mechanisms. The TEDG configuration and the names used for mapping requests are used in these authorization schemes. A BEA Tuxedo client request to a service name offered by the TEDG (`DM_REMOTE_SERVICES`) is subjected to standard BEA Tuxedo authorization by the TEDG in the same way that each server in a secure system is authorized. The TEDG then forwards the request to the BEA TOP END system, along with the local domain `DOMAINID` as the BEA TOP END client user ID. This request is not subjected to further security because the administrator has connected the BEA Tuxedo and BEA TOP END systems in a trusted relationship. When enqueuing requests to a BEA TOP END RTQ queue in a secure BEA TOP END system, the TEDG provides its local domain `DOMAINID` as the BEA TOP END client ID and its configured password (refer to the `dmadmin topendpasswd` command) as part of the request so that those client credentials can be used by RTQ when it dequeues the request and schedules shipment of it to a server.

See Also

- `dmadmin(1)` in the *BEA Tuxedo Command Reference*
- `topendpasswd(1)`in the *BEA Tuxedo Command Reference*

How to Establish Security for BEA TOP END Services/RTQ Queues

To secure a BEA TOP END service or RTQ queue offered by the TEDG, you must secure the BEA Tuxedo service (`SERVICE` or `QSPACE`) name assigned by the TEDG configuration using BEA Tuxedo security tools. To secure a BEA TOP END service that will receive requests from a BEA Tuxedo system through the TEDG using RTQ, secure the BEA TOP END product/function for that service using BEA TOP END security tools and authorize the TEDG local domain ID to access it.

How BEA TOP END Client Requests Are Authorized

A BEA TOP END client request to a product/function or RTQ group/queue offered by the TEDG (`DM_LOCAL_SERVICES`) is subjected to the standard BEA TOP END authorization by the BEA TOP END client's system. The TEDG then subjects it to an optional access check based on the `ACL` parameter and the remote domain `DOMAINID` as configured in the `DMCONFIG` file. It then forwards the request to the BEA Tuxedo system along with the remote domain identifier of the BEA TOP END node as the client ID. Depending on the BEA Tuxedo security level (that is, if `ACL` and `MANDATORY_ACL` are being used), this request is then subjected to BEA Tuxedo authorization that takes place in the server application. Security associated with the dequeuing or forwarding of messages from a BEA Tuxedo /Q is unaffected by the TEDG.

How to Establish Security for BEA Tuxedo Services/Queue Spaces

To establish security for a BEA Tuxedo service or queue space that will receive requests from a BEA TOP END system through the TEDG, use BEA Tuxedo security tools and authorize the BEA Tuxedo remote domain `DOMAINID`(s) to access the service or queue space. Follow this procedure for systems in which security is provided by the `ACL` or `MANDATORY_ACL` method. Use BEA TOP END security tools to authorize the BEA TOP END client to access:

- The product and function that provides the target BEA Tuxedo service.
- The RTQ group and queue name associated with the BEA Tuxedo queue space specified in the TEDG `DMCONFIG` file.

How Security Is Provided for the TEDG Network Connection

The TEDG network connection to the BEA TOP END system is the final element of end-to-end security with the TEDG. If security is configured in the BEA TOP END system, this connection is authenticated based on the BEA TOP END internode security protocol and the credentials installed for the system in the `srvtab` file. Additionally, based on the configuration level of the TEDG and the BEA TOP END system, link-level encryption is provided for messages passed through the link based on the BEA TOP END internode security protocol. BEA Tuxedo link encryption protocols are not used for the TEDG.

See Also

- "Understanding Security Between the BEA TOP END and BEA Tuxedo Systems" on page 3-1
- "Configuring Security Between BEA TOP END and BEA Tuxedo Systems" on page 7-1